ALL THE KING'S HORSES:

The Delta Wing Leading-Edge Vortex System Undergoing Vortex Breakdown: A Contribution to its Characterization and Control under Dynamic Conditions.

By
Norman W. Schaeffler

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
IN
ENGINEERING MECHANICS

APPROVED:

Demetri P. Telionis, Chair

Roger L. Simpson — Ronald D. Kriz

Muhammad R. Hajj — Dean T. Mook

April 20, 1998
Blacksburg, Virginia
The United States of America

Key Words: Delta Wing Aerodynamics, Vortex Breakdown, High Angle of Attack Control

ALL THE KING'S HORSES:

THE DELTA WING LEADING-EDGE VORTEX SYSTEM UNDERGOING VORTEX BREAKDOWN: A CONTRIBUTION TO ITS CHARACTERIZATION AND CONTROL UNDER DYNAMIC CONDITIONS

By

Norman W. Schaeffler

Demetri P. Telionis, Chairman
Department of Engineering Science and Mechanics

The quality of the flow over a 75°-sweep delta wing was documented for steady angles of attack and during dynamic maneuvers with and without the use of two control surfaces. The three-dimensional velocity field over a delta wing at a steady angle of attack of 38° and Reynolds number of 72,000 was mapped out using laser-Doppler velocimetry over one side of the wing. The three-dimensional streamline and vortex line distributions were visualized. Isosurfaces of vorticity, planar distributions of helicity and all three vorticity components, and the indicator of the stability of the core were studied and compared to see which indicated breakdown first. Visualization of the streamlines and vortex lines near the core of the vortex indicate that the core has a strong inviscid character, and hence Reynolds number independence, upstream of breakdown, with viscous effects becoming more important downstream of the breakdown location. The effect of cavity flaps on the flow over a delta wing was documented for steady angles of attack in the range 28° to 42° by flow visualization and surface pressure measurements at a Reynolds number of 470,000 and 1,000,000, respectfully. It was found that the cavity flaps postpone the occurrence of vortex breakdown to higher angles of attack than can be realized by the basic delta wing. The effect of continuously deployed cavity flaps during a dynamic pitch-up maneuver of a delta wing on the surface pressure distribution were recorded for a reduced frequency of 0.0089 and a Reynolds number of 1,300,000. The effect of deploying a set of cavity flaps during a dynamic pitch-up maneuver on the surface pressure distribution was recorded for a reduced frequency of 0.0089 and a Reynolds number of 1,300,000 and 187,000. The active deployment of the cavity flaps was shown to have a short-lived beneficial effect on the surface pressure distribution. The effect on the surface pressure distribution of the varying the reduced frequency at constant Reynolds number for a plain delta wing was documented in the reduced frequency range of 0.0089 to 0.0267. The effect of the active deployment of an apex flap during a pitch-up maneuver on the surface pressure distribution at Reynolds numbers of 532,000, 1,000,000, and 1,390,000 were documented with reduced frequencies of 0.0053 to 0.0114 with flap deployment locations in the range of 21° to 36°. The apex flap deployment was found to have a beneficial effect on the surface pressure distribution during the maneuver and in the post-stall regime after the maneuver is completed.

TABLE OF CONTENTS

LIST OF FIGURES AND MEDIA OBJECTS

NOMENCLATURE

C_p	Pressure Coefficient, $(p-p_\infty)/(\frac{1}{2}\rho U_\infty^2)$
C	Chord length of model
C'	Projection of the chord length ($C'=C\cos\alpha$)
h	Spacing between pressure taps (Chapter 5)
k	Reduced Frequency, $\frac{\dot{\alpha}C}{2U_\infty}$
p	Model surface pressure
p_∞	Free-stream static pressure
Re	Reynolds number based on chord length
S	Total length of local half span (Chapters 4 and 5)
U_∞	Free-stream velocity
x	Distance from centerline along local half span (Chapters 4 and 5)
y	Distance measured along centerline of wing (Chapters 4 and 5)
X	Distance from apex in the streamwise direction (Chapter 3)
α	Angle of Attack
$\dot{\alpha}$	Angular Pitch Rate
Λ	Sweep angle of the wing
ρ	Density of air

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank those who have helped me to complete this work, for I did not do it alone.

My heartfelt appreciation goes out to Dr. Demetri Telionis for making it possible for me to return to Blacksburg and to pursue a Ph.D. Without his guidance and mentoring, this work would not exist. I will be forever grateful for the opportunities that he has given me.

I would like to thank Dr. Ron Kriz for introducing me to the power of visual computing and to the idea that a set of connected computers has a power far greater than the sum of the individual machines. I would also like to thank him for his enthusiasm about my work and for being the one out there fighting to build the world-class facilities that I had the good fortune to be able to use.

I would like to thank Dr. Roger Simpson for giving me the opportunity to work with the DyPPiR as intimately as I have. I would also like to acknowledge his assistance and advice in times of experimental frustrations. I am grateful for both.

To Dr. Dean Mook and Dr. Muhammad Hajj, the last two members of my committee, thank you for your constructive comments and support throughout the life of this work. Your input was greatly appreciated.

This work was supported by the United States Air Force Office of Scientific Research, Project Number AFSOR-91-0310 and AASERT-F49620-93-1-0455 and was monitored by Major Daniel Fant and later by Dr. James McMicheal. Their support is gratefully acknowledged.

I would like to acknowledge the support of my family, who never once questioned the wisdom of quitting my “real job” and returning for yet more schooling. My parents, Norman and Judi Schaeffler, my in-laws, Bernard and Carlyn Miller, my sister Cathy and her husband

Will Smith and my brother-in-law, Paul and his wife Tammy Miller. Thanks to you all for the support you have given me over the years that it took to complete this work.

I would also like to acknowledge the people who have made these last few years more enjoyable. Friends, old and new. Othon and Debby Rediniotis, Ngoc and Lan Hoang, Matt and Polly Zeiger, Mike Wilder, Martin and Keri Donnelly, Andy Mathes, Chris Moore, Erik and Aleta Panzer, Luis Chalmeta, Pavlos Vlachos, and Dimitri Stamos. And last but not least, Lucinda Willis, my coffee-drinking buddy. Thank you for being a constant source of support and always reminding me to "stop and smell the roses".

Finally, the completion of my degree would not have been possible without the loving support of my wife, Louise. She has made tremendous sacrifices over the years in order for me to pursue the dream of obtaining my Ph.D. It is at least as much hers as it is mine. I thank you from the bottom of my heart for all that you have done.

Dedicated to Louise and Carlyn

CHAPTER 1: INTRODUCTION

When a uniform stream encounters a delta wing at a positive angle of attack, the flow attaches to the windward side of the wing. A line of attachment is formed coincident with the centerline of the wing and the flow is diverted either to port or starboard. Boundary layers develop on the windward side of the wing, originating at the line of attachment and developing as the fluid moves towards the leading edge. Upon reaching the leading edge, the boundary layers, unable to negotiate the sharp corner of the wing, separate and form two free-shear layers. These free-shear layers in turn, organize themselves on the leeward side of the wing into a symmetric pair of counter-rotating vortices. The existence of these two vortices is the essence of the delta wing flowfield. The vortices induce axial velocities within their cores on the order of two to three times the free-stream velocity and support circumferental velocities approaching two and a half times the free-stream velocity. These large axial velocities generate an incremental lift for the wing, usually referred to as vortex or non-linear lift. The vortex strength and hence, the axial velocity induced in the core, increases as the angle of attack increases, but only up to a point. Above a critical angle of attack, a fundamental change in the structure of the vortex occurs and the high axial velocities within the core can no longer be sustained. The axial velocity decreases, the vortex grows in diameter and the circumferental velocities correspondingly decrease. The vortex has “broken down”.

1.1 Delta Wing Aerodynamics

The typical airframe application of the delta wing is the jet fighter. The requirements for a high-performance “supermaneuverable" fighter aircraft dictate a blend of high supersonic cruise ability and optimal low speed control. It is for the former reason that the delta wing is the planform of choice. The latter requires a wing with excellent low Mach number flight characteristics, a well-known weakness of delta wings. The presence of the

vortices on the leeward side of the wing do enhance the aerodynamic characteristics of the delta wing over that which could be expect if the flow was to remain attached to the wing. The non-linear character of the vortex lift that is produced by the axial velocities in the cores of the vortices can easily be seen in Figure 1.1. In this figure, the coefficient of lift is presented as a function of the angle of attack. By comparing the results from the linear theory, i.e. the result if the flow did not separate, to the results for the delta wing, the impact of the vortex-induced lift is apparent. The lift generated by the delta wing is as much as 64% greater than that predicted by linear theory. It can also be seen that at an angle of attack of 32°, the coefficient of lift begins to decreases as the angle of attack increases. This marks the beginning of the occurrence of vortex breakdown over the planform of the wing. The occurrence of vortex breakdown over a delta wing at high angles of attack is cause for concern for an airframe designer attempting to utilize a delta wing in their design.

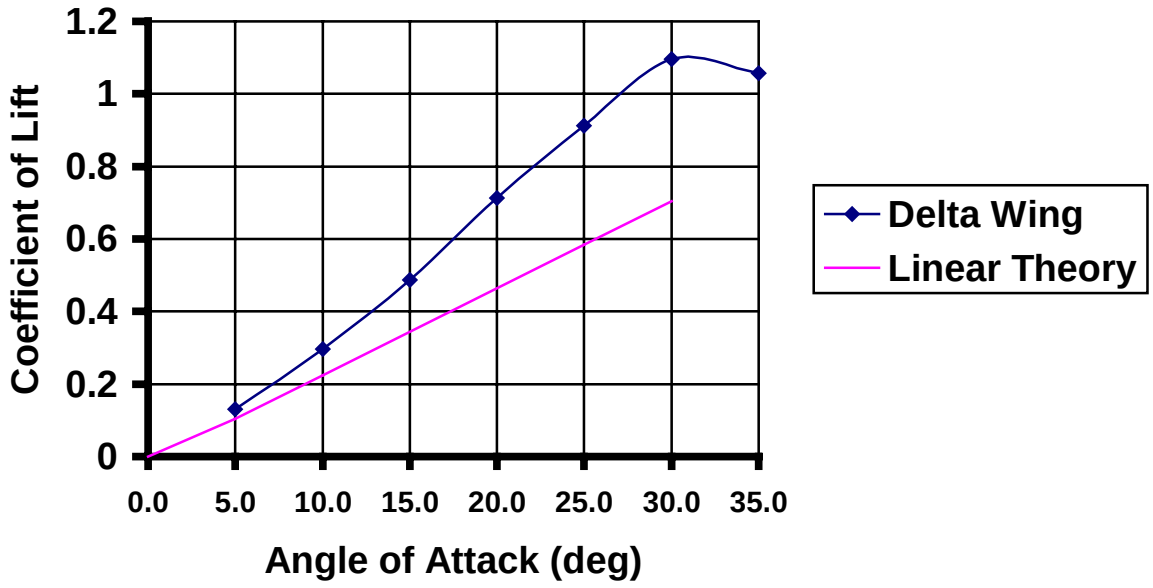


Figure 1.1: The Coefficient of Lift versus Angle of Attack for a 76°-sweep angle delta wing and the corresponding results from linear theory. Data is from Peckham and Atkinson (1957)

1.2 Vortex Breakdown

The flow over a delta wing is just one example of a flow where vortex breakdown is observed. Vortex breakdown is also observed in swirling flows in tubes (Sarpkaya, 1971, 1995), in tornado-like flows, free swirling jets, in trailing vortex systems, in rotors, and in combustion chambers (Visbal, 1995). Of all these, the case of swirling flows in tubes has been the most frequently studied and most often emulated by theoreticians. Experimental studies of columnar vortex breakdown have been carried out as described in the detailed reviews of Wedemeyer (1982), Leibovich (1984) and Escudier (1988). Based on his study of swirling flow in a tube, Sarpkaya (1971) classified three type of vortex breakdown, the bubble, the spiral, and the double helix. Upon revisiting the subject, on the same experimental apparatus, Sarpkaya (1995) added a fourth form, which he termed conical. The four types of vortex breakdown as defined by Sarpkaya are shown in Figure 1.2. Sarpkaya noted that during the transient adjustment period which follows a change in a flow parameter, the breakdown may move from one type to another, e.g. from a spiral to a bubble and then back to a spiral. Noting a similar progression, Faler and Leibovich (1978), using an apparatus identical to that of Sarpkaya, added two other classifications to include some of the transients forms, bringing the total number of vortex breakdown types to six.

The breakdown of leading-edge vortices over delta wings has been documented via flow visualization by a number of investigators (Lambourne and Bryer 1961, Payne, et al. 1988). However, quantitative measurements are rather limited and most of them are confined to surface pressures or limited velocity measurements (Roos and Kegelman, 1990, Atta and Rockwell, 1990). Detailed two-dimensional velocity fields were presented using laser-Doppler velocimetry and seven-hole probes (Meyers and Hepner, 1988, Payne, et al. 1989, Kegelman and Roos, 1990, Rediniotis et. al., 1991) or particle-image velocimetry (Magness et. al., 1993).

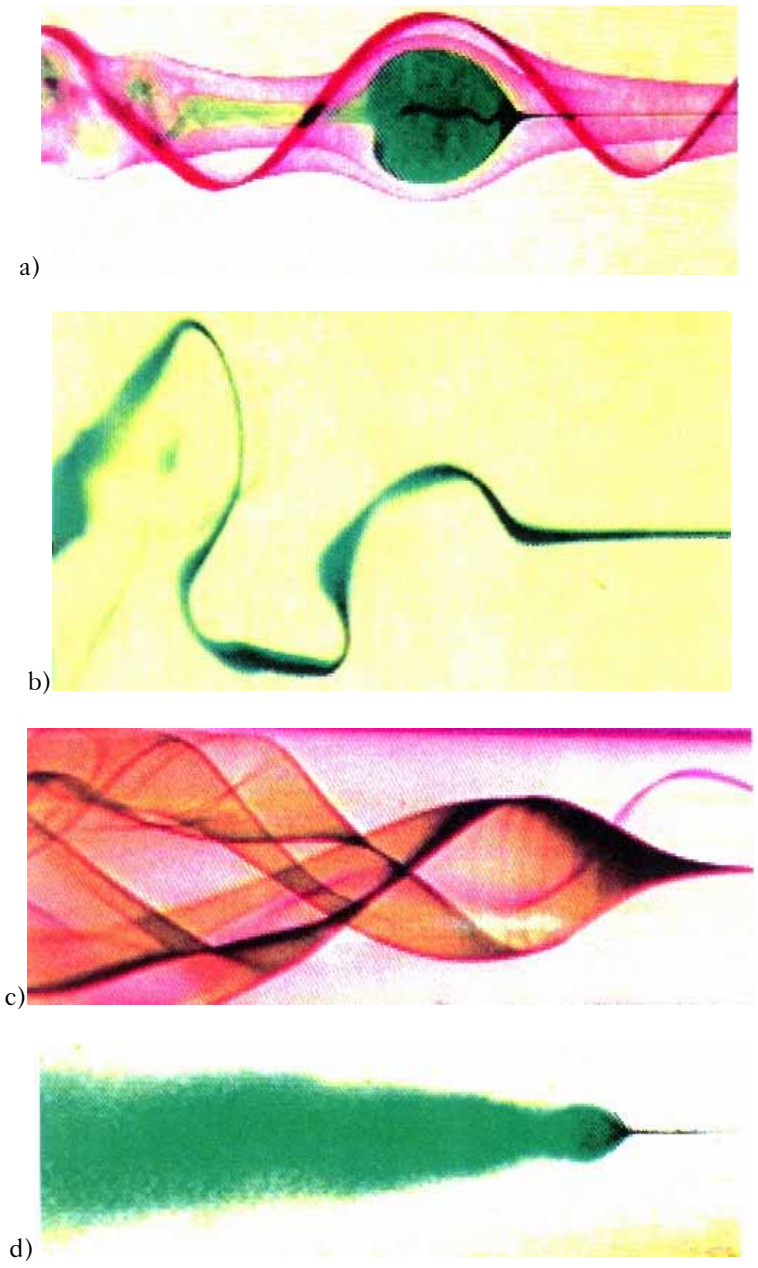


Figure 1.2: The four types of vortex breakdown as defined by Sarpkaya. a) bubble, b) spiral, c)double helix, and d) conical. Vorticies are visualized by the use of dye. Photographs are from Sarpkaya (1994).

While the occurrences of vortex breakdown in different applications share similar features, there are certain characteristics that are specific to each of the flowfields. The need to understand, predict, and model vortex breakdown has lead to investigations on the structure of vortex breakdown specifically as it occurs over the delta wing. Of the vortex breakdown types enumerated by Sarpkaya, the bubble and spiral types are most commonly seen in flow visualizations over delta wings at low Reynolds number. Flow visualization has been the prime vehicle for examining the structure of vortex breakdown over a delta wing. As a result, several features of the delta wing flowfield, with and without vortex breakdown, have been noted in the literature.

Among these is that the shear layer that feeds the leading-edge vortex also forms two vortex substructures. The first of these two structures is due to an instability of the shear layer. This instability is temporally periodic and appears in flow visualizations to form in lines parallel to the leading edge. Gad-el-Hak and Blackwelder (1985) first observed this instability. They observed vortex pairing within the shear layer as it rolled into the primary vortex. Based on these observations, Lowson (1989) attributes this instability to a mechanism similar to the two-dimensional Kelvin-Helmholtz shear layer instability. It is not known if this instability can be observed at high Reynolds number or if the presence of vortex breakdown has any effect on it.

The second substructure is a vortex-cell type instability, which is spatially stationary. These structures wrap around the primary vortex in a helical fashion maintaining their own integrity instead of merging into the primary vortex. They have been previously observed by Payne (1987), Lowson (1989), and Washburn (1992). A detailed study of these structures at Reynolds numbers of 1,250,000 and angles of attach up to 25° was performed by Washburn and Visser (1994). The role of these structures in vortex breakdown and their existence at high angles of attack has not been previously documented, but its existence at a higher angle of attack will be shown here.

The application of some topological concepts to the flow over a delta wing has proven insightful. Inspired by the work of Tobak and Peake (1982) and Hunt, et. al. (1978), the sectional streamline patterns in a plane perpendicular to the wing can be given a topological interpretation as a connected set of saddles and nodes. The stability of the nodes, which are the vortex cores, turns out to be of prime importance. For a stable node, the streamlines spiral towards the node (core), and for an unstable node, the streamlines spiral away from the node. The criterion for determining whether or not a node is stable or not is the sign of the trace of the two-dimensional rate-of-deformation tensor in the neighborhood of the node. The trace is nothing more than the divergence of the velocities in the cross-flow planes. If the trace is positive, the node is stable. If the trace is negative, the node is unstable (Perry and Chong, 1987). For a delta wing in pitch-up motion, Magnes et. al. (1993) reported that the vortex center was unstable downstream of the breakdown location. Visbal and Gordnier (1994) demonstrated numerically that downstream of the breakdown location, the node is unstable for the case of a stationary wing.

Finding the stability of the vortex core in a topological sense can be used as an additional criterion for determining that vortex breakdown has begun. Of all of the criterions that are available, which is most sensitive? Which indicates first that breakdown is eminent? Traditionally, it has been the distribution of axial velocity and axial vorticity that are offered as indicators that breakdown has begun. (Rediniotis, et. al. 1991). The presence of a point within the vortex core with zero axial velocity has also been used as an indicator. The theoretical work of Brown and Lopez (1993) suggest that the azimuthal vorticity changing sign from positive to negative is the governing criterion for inducing breakdown, therefore the occurrence of this sign change within the vortex would indicate that the breakdown process has begun. The helicity distribution within the vortex can also be used as an indicator of the start of the vortex. The use of both of these as an indicator will be examined here.

1.3 The Control and Management of Vortex Breakdown during Dynamic Maneuvers

The control of a delta wing aircraft at high angle of attack is basically an exercise in the control of the two counter-rotating vortices that exist on the leeward side of the wing. It is desirable to keep the vortices as robust as possible to as high an angle of attack as possible. The large suction peaks associated with robust leading edge vortices give pilots ample amounts of aerodynamic "muscle" to control the attitude of their aircraft. Once vortex breakdown has occurred over the planform of the wing, the aircraft has entered the post-stall regime. The two broken-down vortex systems begin to resemble a wake to a higher and higher degree as the angle of attack increases. Hence, the goal of a delta wing control surface system has two objectives. First, postpone the start of the post-stall regime by delaying vortex breakdown to as high an angle of attack as possible and secondly, to offer the ability in the post-stall regime to give some beneficial order to the wake-like broken-down vortex systems.

When a delta wing executes a pitch-up maneuver, there is a lag associated with the appearance of vortex breakdown over the wing relative to the static case. The faster the maneuver, i.e. the larger the value of the reduced frequency of the maneuver, the larger the lag. This has been observed both numerically (Visbal and Gordnier, 1994) and experimentally (Thompson, et. al., 1991). In addition, for the dynamic case, once breakdown appears over the wing, it appears uniformly all along the chord length of the wing (Rediniotis et. al., 1994). This is in contrast to the steady case, where the breakdown position moves progressively up from the trailing edge. As Rockwell (1993) points out, any flow control device, which is abruptly turned on during a maneuver, will have an effect that has its own lag associated with it. This makes the proper timing of a control surface deployment more difficult to manage. The lag in the appearance of vortex breakdown associated with the pitching of the wing itself is a function of the reduced frequency and once breakdown is over the wing it spreads over the entire chord quickly.

Several control surface configurations have been investigated and reported in the literature. Among these, leading-edge flaps, also referred to as leading-edge vortex flaps, have received considerable attention. In this control surface configuration, the entire leading edge of the wing is hinged to deflect up or down. When deflected down, the effect is to have the leading edge vortex reattach to the flap itself and not the wing. This decreases the effective angle of attack, which is advantageous at high angles of attack. When deflected up, the angle of attack appears larger, thus increasing lift at low angles of attack. Rao (1979, 1980), Marchman (1981a, 1981b), and Reddy (1981) investigated the effect of these flaps on the performance of delta wings and demonstrated that the two configurations can lead to controlled increases and decreases in drag. They also demonstrated the leading-edge flaps could generate a component of the lift vector in the direction of flight, thus producing a thrusting effect. Gursul et. al. (1995) investigated the use of leading edge flaps and variable sweep wings as a means of controlling vortex breakdown. Lamar (1986) has experimented with a single trailing edge aileron, tip mounted on a cropped delta wing. This was successful in generating a constant roll moment. Broader discussions of advanced control devices can be found in Lamar and Campbell (1983) and in Rao and Campbell (1987).

Another type of control surface is the apex fence. It can be thought of as a leading-edge flap, which has been deflected 90°, but with the flap limited to only the quarter-chord nearest the apex. Hoffler et. al. (1985) claim that this arrangement will increase lift at low angles of attack and has the opposite effect at high angles of attack. At low angles of attack the flap augments lift in the apex region, increasing the positive pitching moment. This can be used to trim the wing if used in conjunction with trailing-edge flaps during take-offs and landings.

A related flap configuration is the tabbed leading edge flap. This is an extension to the traditional leading edge flap with an additional section right at the leading edge which can be deflected back through an angle such that it become parallel to the main portion of

the wing. Hoffler and Rao (1985) found that this configuration tends to increase the flap thrust. It also increases the drag by an equal amount, leaving the lift-to-drag ratio of the wing virtually unchanged. This flap configuration, as well as some of the others discussed here, can be seen in Figure 1.3.

An alternative design of a control surface was introduced and originally investigated by Rao (1985). In this configuration, a flap is extended from underneath the wing, inclined at an angle to the wing, with the leading edge of the flap parallel to the leading edge of the wing. Rao termed this configuration a vortex cavity flap or simply a cavity flap. This configuration is attractive for fundamental work because it does not change the nominal area of the wing and therefore any changes that occur are due only to the deployment of the flap and not to an increase in wing planform. The cavity flap, in steady and dynamic conditions, is one of the central focuses of this work.

The effect of cavity flaps for angles of attack as high as 50° in steady flow was reported by Rao (1987), who demonstrated that the thrust and drag influence of flaps, both leading edge and cavity flaps, remain effective well beyond the stall regime. It is expected that the cavity flap will help modify and control the onset, growth and shedding characteristics of large-scale vortices, but Rao presented data only on the effects of cavity flaps on lift and drag. Surface pressure measurements and velocity field measurements will be presented herein in an attempt to gain a better understand of what is happening in the flowfield as a result of the cavity flaps.

In addition to cavity flaps, an apex flap has been shown to be a useful delta wing control surface (Rao, 1983). An apex flap is implemented on a delta wing by allowing the forward section of the wing to hinge upward or downward relative to the rest of the wing, effectively increasing, or decreasing, the angle of attack of the portion of the wing closest to the apex. Typical angles for the flap to be deployed relative to the wing are on the order of 15°. A typical amount of the wing to deflect is 40% of the chord. For both steady flow and

during a pitch-up motion, continuously deployed apex flaps are capable of extending the maximum angle of attack that can be realized without breakdown occurring over the planform of the wing (Rediniotis et. al., 1993).

1.3.1 Pitch Axis Location during a Maneuver

All the dynamic maneuvers presented here use the apex as the location of the pitch axis. As reviewed by Rediniotis (1992), a number of pitch-axis locations are in common usage in the literature. These include y/C, axis location non-dimensionalized by the root chord, of 0.0, 0.25, 0.5, 0.75 and 1.0. None of these pitch axes locations enjoys universal acceptance. Since the amount of vorticity shed from the leading edge of the wing, as a free-shear layer, is proportional to the square of the velocity at the edge of the boundary layer at the leading edge, pitching about the apex sheds more vorticity than pitching about the trailing edge. However, the axial vorticity distribution, at similar stations down the wing, correspond very well for a case where the pitch is about the trailing edge (y/C=1.0) or about the apex (y/C=0.0) (Rediniotis, 1992). This indicates that the effect of the additional vorticity generated due to the pitch axis location has a small effect on the overall vorticity distribution. Additionally, the location of vortex breakdown is not effected by the pitch axis location. Accordingly, the effect of the pitch axis on the flow physics at work in a dynamic motion of a delta wing is considered to be minimal. Therefore, all the motions documented here will be a ramp pitch-up motion with the pitch axis located at the apex.

1.4 Goals of the Investigation

One of the major goals of this investigation is to better the understanding of the structure of vortex breakdown over a delta wing. A delta wing with a 75°-sweep angle and a aspect ratio 1.07 will be used. This is a highly swept wing more so than is commonly in use on advanced tactical fighters, however it is a sweep angle that is widely documented in the literature. Table 1.1 lists the sweep angle of several modern aircraft.

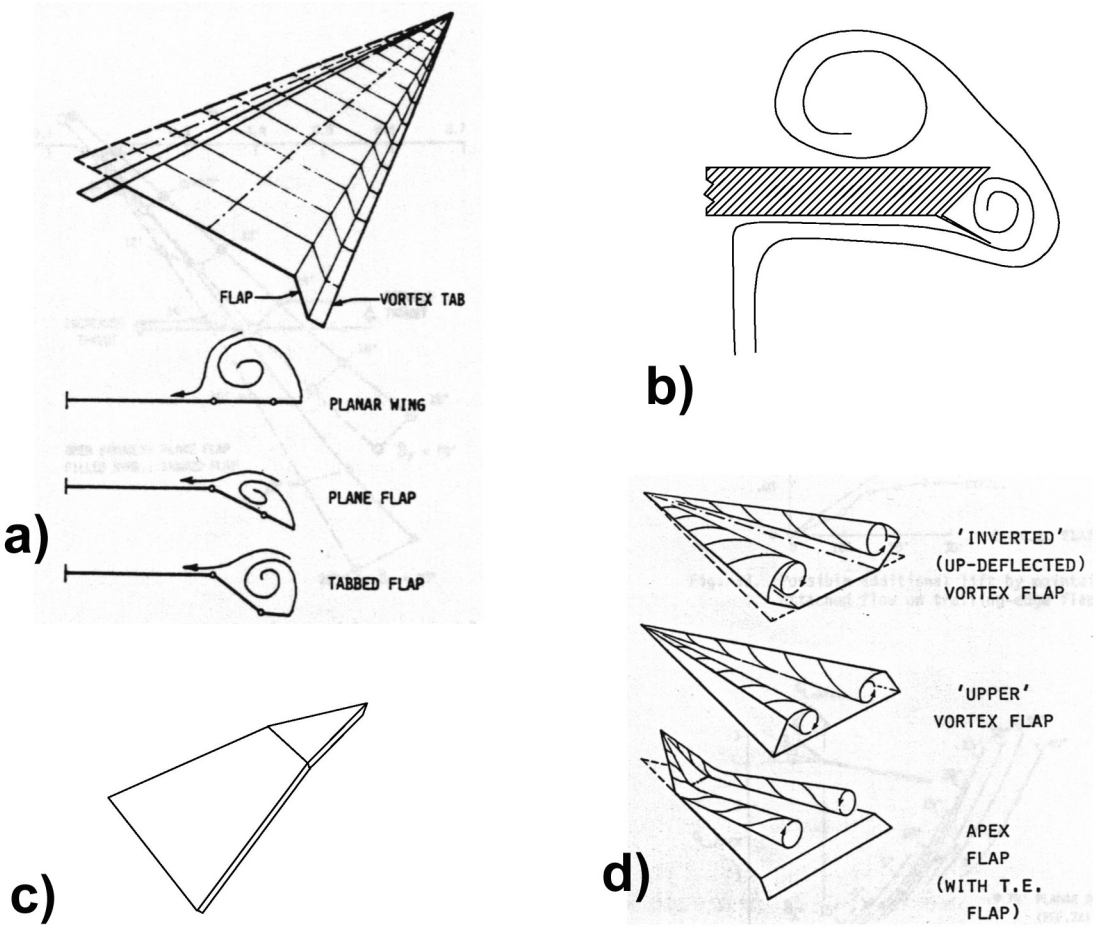


Figure 1.3: A gallery of delta wing control surfaces, a) a tabbed flap (from Hoffler and Rao(1985)), b) cross-section of a cavity flap, c) an downward deflected apex flap, and (d) a inverted vortex flap, a upper vortex flap (allow this only to exist for the fore quarter chord and this is an apex fence), and an upward deflected apex flap and trailing-edge flap (From Rao (1983).

Aircraft	Sweep Angle
General Dynamics F-16XL	70° fore/50° aft
MiG 29KVP	73° 30' on LERX
Lockheed Martin A/F-22X	71° fully swept
Lockheed Martin F-117A	67° 30'
Panavia Tornado MK1A	67° (maximum)
Aerospatiale/BAC Concorde	57°-31° (modified delta)
F-106A	60°
Lockheed F-16	40°
Eurofighter 2000	53°

Table 1.1: Leading-edge sweep angles of various modern aircraft (Jackson, 1995).

The other major goal is to develop techniques for controlling and managing vortex breakdown over the wing during dynamic maneuvers to high angles of attack. To further this goal, the three-dimensional velocity field over the delta wing at a fixed angle of attack (α = 38°) was mapped out using laser-Doppler velocimetry (LDV). At this angle of attack, vortex breakdown occurs about half way down the wing. Using this database for the velocity field, we examined in three-dimensions, streamlines and vortex lines, the spatial distribution of the helicity and the vorticity components and isosurfaces of velocity components, specifically the isosurface of zero axial velocity. Sectional streamline patterns in planes aligned with the free-stream and perpendicular to the wing were also analyzed. The first occurrence of the various breakdown indicators was established and compared with each other and it was determined which provides the first indication that breakdown in eminent.

The effect of two control surfaces on the flow over a delta wing was also examined. First, the effect of the cavity flap was explored. This was done by studying the axial velocity distribution over a delta wing at a fixed angle of attack (α=35°) via LDV. The case where the flaps are not deployed was compared to the case where there are flaps deployed. Also for

steady flow, the surface pressure measurements along lines perpendicular and parallel to the leading edge for fixed angles of attack in the range $28° \leq \alpha \leq 44°$ was examined. Flow visualization using helium bubbles for fixed angles of attack in the range $28° \leq \alpha \leq 44°$, with and without cavity flaps was conducted to gain insight into the modifications to the flow field that the cavity flaps cause.

To ascertain the effects of cavity flaps during a dynamic maneuver, ensemble-averaged surface pressure measurements for a ramp pitch-up motion with flaps deployed throughout the motion were conducted at one Reynolds number for one reduced frequency. Ensemble-averaged surface pressure measurements for a ramp pitch-up motion with flaps deployed at a pre-selected angle of attack during the execution of the motion for two different Reynolds numbers but one reduced frequency were also conducted and the results will be presented herein. Results from a parametric study of the effect of changing the reduced frequency at constant high Reynolds number for a plain delta wing will be presented.

The effect of the apex flap on the flow over a delta wing will be presented in terms of ensemble-averaged surface pressure measurements for a pitch-up maneuver with apex flap deploying at a pre-set angle of attack during the maneuver for three reduced frequencies and two Reynolds numbers. The surface pressure distribution during the motion was recorded and analyzed.

As outlined above, the research presented herein, deals with descriptions of the pressure and velocity fields above a delta wing. It is recognized by the author that what is ultimately of importance is how much lift and drag a wing produces. However, the insight gained by looking in more detail at the flow over the delta wing, instead of concentration on the global integrated effects, will help to clarify the effect that the various control surfaces have on the wing overall.

CHAPTER 2: FACILITIES AND INSTRUMENTATION

The data presented herein were obtained in three separate facilities, two wind tunnels and one water tunnel over five geometrically-similar delta wing models. Since the fluid structure of interest here is a vortical structure, prone to instability, non-intrusive measurement techniques were deemed necessary. The introduction of a mechanical probe into the vortical structure has been shown to influence the behavior of the vortex, in some cases inducing breakdown prematurely (Nelson and Visser, 1982). To facilitate this requirement, two measurement techniques were used throughout to gain insight into the behavior of the vortical structure, laser-Doppler velocimetry (LDV) and surface pressure measurements. In the water tunnel experiments, a two-component LDV system, manufactured by Thermal Systems, Incorporated (TSI), was utilized to make velocity measurements. In the wind tunnels, surface pressure measurements were made using pressure taps on the surface of the models, short tubing runs and a multi-channel electronic scanning pressure transducer, manufactured by Pressure Systems, Incorporated (PSI).

2.1 The Engineering Science and Mechanics Water Tunnel

The Engineering Science and Mechanics (ESM) Water Tunnel was constructed in 1976 by the Engineering Science and Mechanics Department, with support of the Army Research Office. The tunnel remained in service until 1994, when it was dismantled and replaced by a larger facility. The data presented herein represents the last extensive set of data obtained in "Nellie Bell", as the tunnel was affectionately known. The tunnel was of a closed circuit design containing 570 gallons of water. It was constructed of plexiglass and poly-vinyl (PVC) pipe. A schematic of the tunnel can be seen in Figure 2.1. This facility has a 0.25-meter by 0.30-meter cross-section and a top speed of 1.5 meter/second as controlled by a centrifugal pump powered by a 2 hp variable speed dc motor. The tunnel featured a 6:1 two-

dimensional contraction and was equipped with screens, honeycombs and a set of flow straighteners and guide vanes in the divergence to improve the flow quality in the test section. Free-stream turbulence levels were 1.0% for free-stream velocities from 12 cm/sec to 25 cm/sec measured at the end of convergence. The turbulence levels within the test section ranged from 1.5% to 5% (Wilder, 1992). A detailed calibration of the tunnel can be found in Koromilas and Telionis, (1980) and in Telionis, et al. (1986). The test section of this facility is equipped with a computer-controlled motor with an optical encoder installed to control the angle of attack, to within an accuracy of 0.1°. A detailed description of this feature of the facility can be found in Rediniotis (1992).

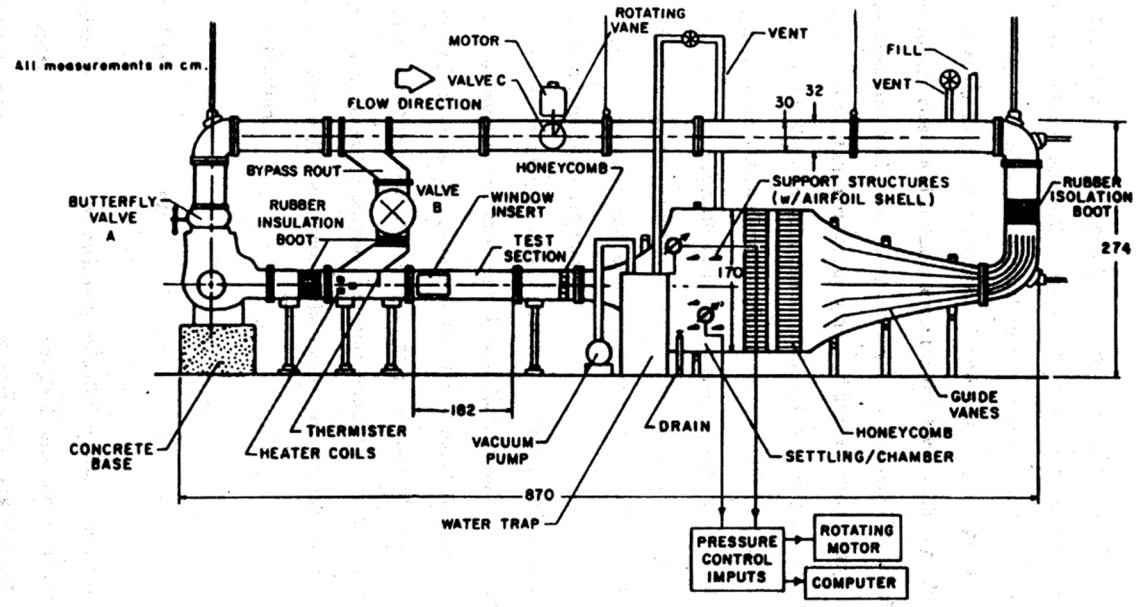


Figure 2.1: The Engineering Science and Mechanics Water Tunnel (Wilder, 1992)

2.2 The Engineering Science and Mechanics Wind Tunnel

The Engineering Science and Mechanics (ESM) Wind Tunnel was constructed in 1983 and is a low speed wind tunnel of open circuit design. The speed of the tunnel can be adjusted by a changing the diameter of the drive pulley on the DC motor, which provides the power for the tunnel. In its current configuration, the tunnel can achieve a maximum dynamic pressure of 0.75 torr, which corresponds to a free-stream velocity of approximately 13.3 m/s. The contraction ratio of the tunnel is 5.2:1 and the test section measures 51 by 51 by 125 cm. The maximum free-stream turbulent intensity is 1.25% (Seider, 1983). The ESM Wind Tunnel is shown schematically in Figure 2.2.

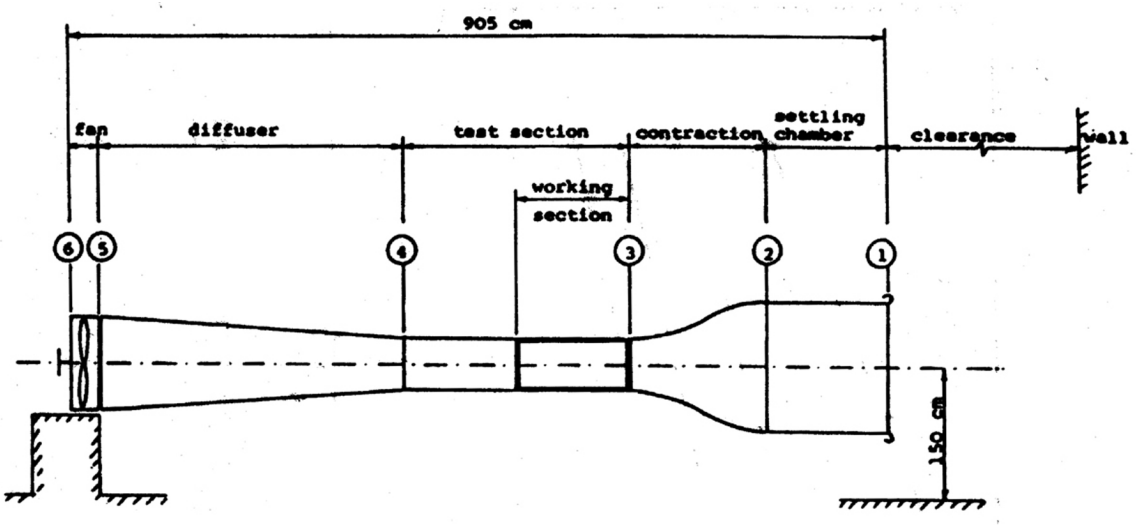


Figure 2.2: The Engineering Science and Mechanics Wind Tunnel (From Seider, 1983)

2.3 The Virginia Tech Stability Wind Tunnel

The Virginia Tech Stability Tunnel is a subsonic, continuous-circuit, single return wind tunnel with a six-foot by six-foot test-section and an excellent quality of flow. Powered by a 600 hp DC motor with 14 foot propellers, the tunnel reaches a top speed of 275 feet per

second. A photograph of the Virginia Tech Stability Tunnel is presented as Figure 2.3. The Virginia Tech Stability Wind Tunnel was built in 1940 at the Langley Aeronautical Laboratory (Langley Field) which was run by the National Advisory Committee on Aeronautics (NACA), predecessor to today's National Aeronautics and Space Administration (NASA). Today, the Langley Aeronautical Laboratory is known as the Langley Research Center. The tunnel was constructed by NACA to allow the experimental determination of dynamic stability derivatives and became known around Langley as the "stability tunnel", this moniker eventually became the official name of the tunnel. The design and construction of the tunnel is typical of the tunnels built at Langley at this time. Several tunnels that bear a "architectural" resemblance to the Stability Tunnel are still in use at NASA Langley today. Virginia Tech acquired the Stability Tunnel in 1958 from NASA Langley. The tunnel was re-built at its present location adjacent to Randolph Hall in 1959. The Virginia Tech Stability Tunnel has been full operational since 1961.

The installation by NACA of seven anti-turbulence screens, coupled with the tunnel's 9:1 contraction ratio, gives the Virginia Tech Stability Wind Tunnel an excellent quality of flow. Over a wide range of free-stream velocities, the turbulence intensities are less than 0.05%. In the speed range that the current research was run, the turbulence intensity is less than 0.03%. A slight negative pressure gradient exists in the test section, $\partial C_p/\partial x = 0.33\%$ per meter (Rediniotis, 1992). The setting chamber of the tunnel is 3-meter long and the diffuser has an angle of 3°.

In 1994, the tunnel motor was completely overhauled and restored to original condition. In 1996, the fan blades were completely reconditioned. The current research was

Figure 2.3: The Virginia Tech Stability Wind Tunnel.

conducted around and following these events. The Virginia Tech Stability Wind Tunnel is shown schematically in Figure 2.4.

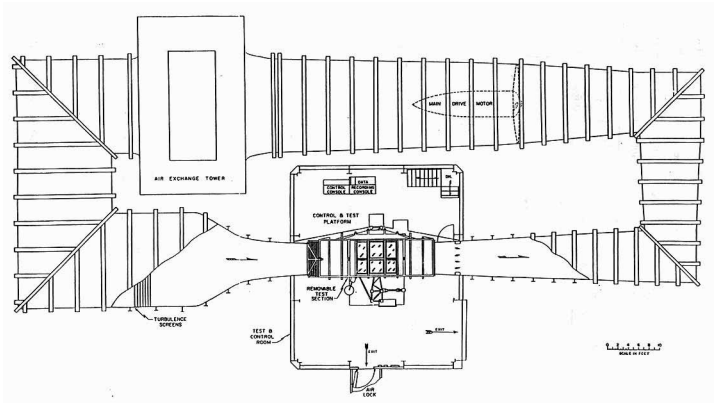


Figure 2.4: Schematic of the Virginia Tech Stability Wind Tunnel. (From Aerospace and Ocean Engineering, 1997).

2.3.1 The Dynamic-Plunge-Pitch and Roll Model Mount System

The Virginia Tech Stability Wind Tunnel is home to a dynamic model mount system that is called the Dynamic-Plunge-Pitch and Roll (DyPPiR) model mount system. The DyPPiR can provide simultaneous plunging, pitching and rolling of models on the order of 150 lb. in weight, at frequencies of up to 10 Hz, depending on the amplitude of the motion. The DyPPiR computer independently controls each of these three motions. Any combination of arbitrary motions is possible. A schematic of the DyPPiR is shown in Figure 2.5. The DyPPiR consists of a carriage, which is constrained by two round-way bearings to run along a track housed in a vertical strut. Mounted on one end of this carriage is a pitch actuator. The pitch actuator has two arms which extend around a roll actuator. A sting is mated with the roll actuator and the model is mounted at the other end of the sting. The entire carriage, with the pitch and roll actuators, sting, and model are moved up and down by the plunge actuator. Therefore, the angle of attack and roll relative to the carriage, and thus the tunnel, is controlled by the pitch and the roll actuators, while the elevation of the model is controlled by the plunge actuator.

The DyPPiR is powered by a high-pressure (3,000 psi) electrohydraulic control system. The plunge actuator is capable of six feet of travel and is situated underneath the test section, between twin concrete columns, which forms the top of the foundation of the DyPPiR. These columns and some of the hydraulic support equipment for the DyPPiR can be seen in Figure 2.6. The plunge actuator is a heavy actuation unit, capable of driving the carriage and the model, which together can weigh over 1000 pounds, with great force. Plunging and pitching oscillations at frequencies of up to 7 Hz are possible for amplitudes of ±16” and ±45° respectively. Rolling oscillation of up to 5 Hz at an amplitude of ±22.5° can also be achieved. The maximum actuator speed is 30 ft/sec for the plunge, 8 rad/sec for the pitch, and 6 rad/sec for the roll.

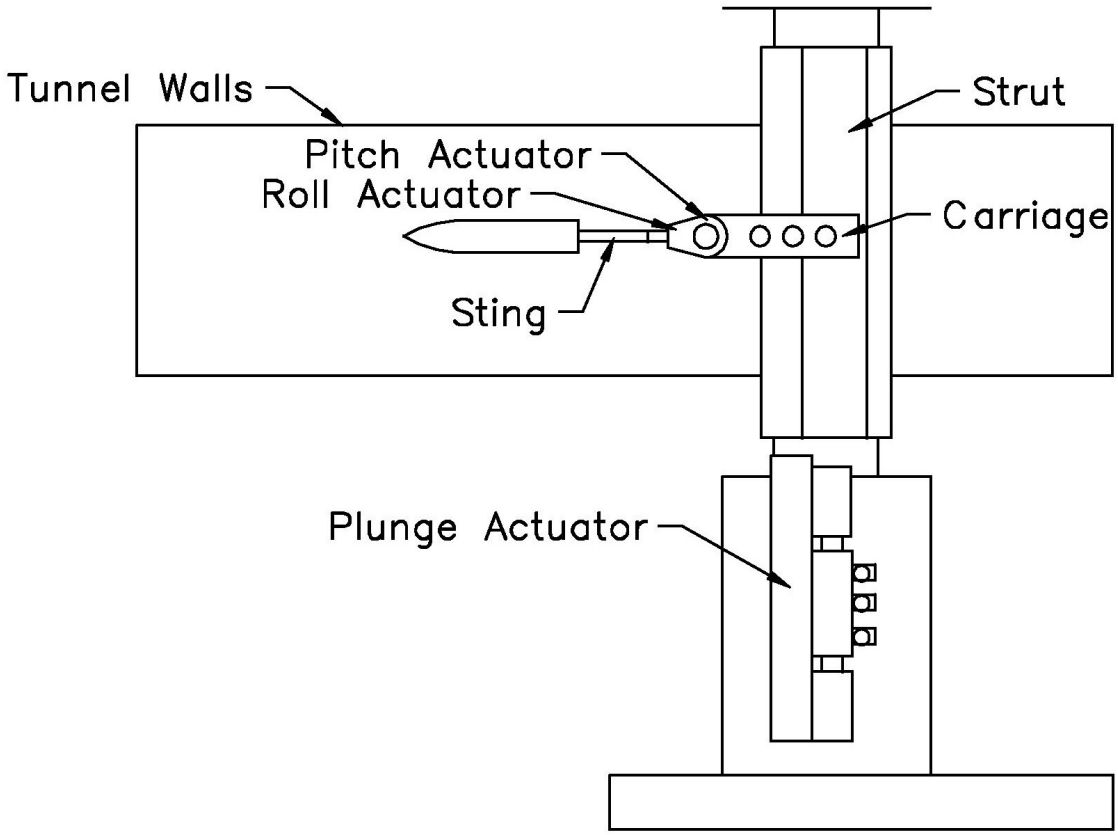


Figure 2.5: Schematic of the Dynamic-Plunge-Pitch and Roll Model Mount System as installed in the Virginia Tech Stability Wind Tunnel.

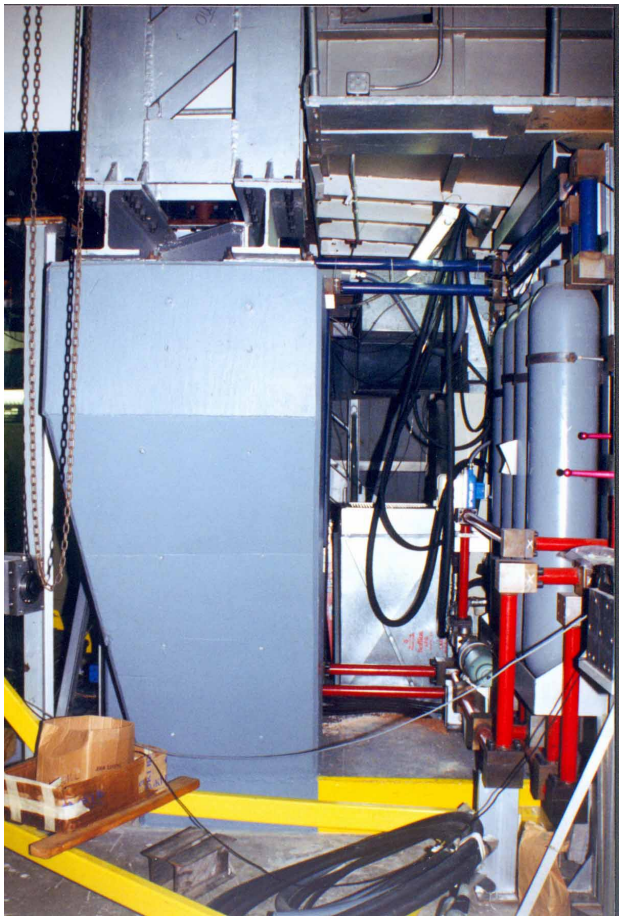


Figure 2.6: The foundation of the DyPPiR. The plunge actuator is housed between the two concrete pylons. The large tanks to the right are accumulators for the hydraulic system. The test section is at the top of the photograph.

The motions that the DyPPiR executes are controlled by a personal computer (PC) and the DyPPiR control electronics. The DyPPiR control computer is equipped with a pair of MetraByte DDA-06 digital-to-analog converter boards. The voltage output from three of the six channels of one of the DDA-06 boards act as command signals for the plunge, pitch, and roll. These command signals are continuously compared against the position signals for each of the degrees of freedom. The plunge cylinder position is indicated by the output from a linear-variable differential transformer (LVDT). The pitch and the roll position are indicated by the output from two resolvers. The DyPPiR control electronics sum the command signal

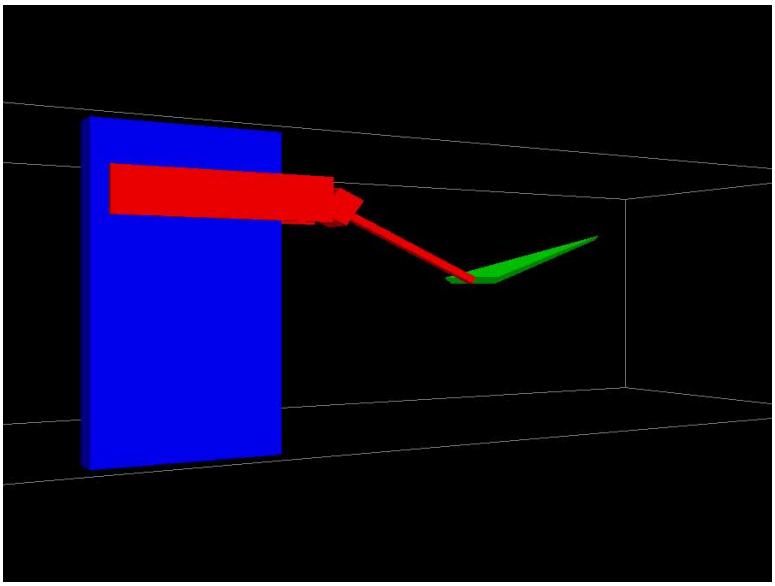
and the negative of the position feedback to generate an error signal. The position of the DyPPiR is then adjusted so as to minimize the error signal. Velocity and acceleration feedback are implemented in the control loops for the plunge and the pitch, in addition to position feedback. The roll utilizes only position feedback. Motion schedules are accomplished by varying the command signals and allowing the DyPPiR control electronics to adjust the DyPPiR position accordingly.

In order for repeatable pitch-up motions of arbitrary speed to be accommodated, the DyPPiR must be capable of executing high-speed maneuvers with high accuracy. The uncertainties of the measured pitch, plunge, and roll positions are 0.053°, 0.023", and 0.164°, respectively (Hoang, et. al., 1994). A discussion of the design, construction, and calibration of this facility can be found in Ahn, et. al. (1989) and the accompanying instrumentation and performance of the DyPPiR can be found in Hoang, et. al. (1994) and Wetzel (1996).

The motions for the DyPPiR to perform are specified in a motion file. Two different motion file formats have been used since the DyPPiR became operational. The original file format was a binary file format, which actually consisted of the voltages that the digital-to-analog (D/A) board in the DyPPiR Control Computer sent as the command signal to the DyPPiR control hardware. This file format proved difficult to work with. Since the raw voltages were stored in the file, the file was dependent on a particular calibration file for the DyPPiR. When the calibration of the LVDT or the resolvers changed, the file now longer specified the intended motion. One advantage of this file format is that all 12 D/A channels were exposed to the user in the motion file. This made the adding of trigger lines, such as is required to deploy a flap at a certain angle of attack, very easy. A second file format was created to overcome the problem of a motion file being tied to a single calibration file. In this file format, the plunge, pitch, and roll positions as a function of time are stored as actual dimensional values, e.g. 25° for a pitch position. The coordinates are translated into voltage as the file is read into the control program. This allows a single motion file to remain forever

valid. Since access to the raw voltages sent out of the D/A board was lost, a different technique was required to deploy the flaps with the new motion file format, dubbed the General Motion File format or the GMF format. The new system involved using a hardware counter to count a clock train from the DyPPiR control computer. This clock train was in sync with the D/A conversions of the command signals. The counter was pre-set with a value and triggered the flaps once that count was met.

So the reader can gain a better understanding of the physical arrangement of the DyPPiR, Media Object 1.1 presents a computer-generated image of the DyPPiR, which is from a piece of software used to test motions for the DyPPiR, the DyPPiR Simulator. The image is a link to a Quick Time Virtual Reality (QTVR) movie of the DyPPiR as it appears in the DyPPiR Simulator.



Media Object 2.1: The DyPPiR as seen in the DyPPiR Simulator used to test motions. The blue rectangle is the pylon, the red objects are the carriage and sting, and a green delta wing of 1.00-meter chord is attached at a 50° offset. Grey lines represent the bounds of the tunnel. All objects are drawn to scale. Click the image above to access a QuickTime Virtual Reality (QTVR) movie of the DyPPiR Simulator. Click here to see the DyPPiR execute a manuever.

2.4 The TSI LDV System

The TSI LDV system used in this investigation is a two-component system built around a 135 mW Helium-Neon (He-Ne) laser. The optical train features a beam collimator, tow polarization axis rotators, a pair of beam splitters, two Bragg cells, beam steering wedges, single photomultiplier, and a beam expansion unit with a 2.27 expansion unit. The beam expander reduces the measurement volume length by 5 times and its diameter by 2.27 times, improving the signal-to-noise ratio (SNR) by 5 times. The final measurement volume dimensions are 1.26 mm in length and 0.089 mm in diameter, with a fringe spacing of 4.5 μm. Four beams are created from the single laser beams to form a measurement volume capable of measuring two velocity components. Frequency shifting of 40 and 60 MHz by the Bragg cells is used to allow the signals for both of the velocity components to be received by a single photomultiplier. The beams are paired so that one velocity component is measured from the crossing of a 40 MHz shifted beam and a 60 MHz shifted beam. Thus the measured frequency of the scattered light is shifted from the Doppler frequency by an amount of 20 MHz. The other velocity component is measured from the crossing of a 40 MHz shifted beam and an unshifted beam, resulting in a measured frequency of the scattered light of the Doppler frequency plus 40 MHz . This allows the total signal from the photomultiplier to be electronically filtered, using band-pass filters, resulting in two signals. Each of these signals is electronically down-mixed with its respective Bragg cell frequency before being passed to the LDV signal processors. Additional frequency shifting is done to the laser beams at approximately twice the mean Doppler frequency of the flow to allow the detection of negative velocities. This additional frequency shift is not down-mixed from the signal. A particle with zero velocity will produce a measured frequency equal to the shift frequency.

The electronically separated output from the photomultiplier was sent to the LDV signal processors, in this case, a set of TSI counters (Models 1980 and 1990). Signal validation was accomplished in a non-coincidence N-cycle mode, using N=8 cycles. In this mode, the counter counts eight cycles, or occurrences where the LDV signal crosses a preset

amplitude threshold eight times, to validate the signals. This translates into the fact that only those particles that cross eight or more fringes in the measurement volume can contribute valid signals. Further signal validation is accomplished by comparing the transit time for the first four cycles to the transit time for the second four. If these two times are not within 7% of one another, the sample is rejected. A hysteresis limit is also used to avoid the timing of turbulent fluctuations, rather than proper cycles of the signal. The TSI counters are equipped with a front panel accessible analog output. This analog output is a voltage, which is proportional to the measured frequency. The constant of proportionality from voltage to frequency is determined by the front panel settings of the counters and is provided by TSI as function of those settings. This voltage was sampled by a computer-based data acquisition system.

Seeding was added to the water tunnel in the form of silicon carbide (TSI Part #900810). Silicon carbide has a mean diameter of 1.5 μm, a density of 3.2 gm/cm^3 and a refractive index of 2.65. Seeding was added to the tunnel every 12-24 hours or when the validated data rate fell below 500 validated data points per second.

The entire data acquisition process was fully automated through an IBM PS/2 Model 55 computer and an analog-to-digital (A/D) and digital-to-analog (D/A) converter board (DT2905). The free-stream monitor was a single-component DISA type 55L laser Doppler velocimeter. The LDV system can be set-up such that the laser beams can be directed into the test section either through the side or the top window of the test section. In this way all three velocity components can be resolved by combining a set of data taken from the side of the tunnel with a set of data for the same grid taken from the top of the tunnel.

Automatic displacement of the measurement volume within a vertical plane is made possible with a pair of stepping motors. Each motor is equipped with a linear variable-differential transformer (LVDT) which acts in a feedback loop with the IBM PS/2 to insure accurate positioning of the measurement volume to within 0.05 mm, (nearly half the

measurement volume diameter). The laser, optics, and vertical and horizontal traversing mechanisms are all mounted on an optical bench, which in turn rests on a traversing platform. The entire system can be traversed manually in the horizontal direction, parallel to the test section.

2.4.1 Uncertainties in the Measurement of the Velocity

In an LDV system, the measurement of the velocity is based upon the measurement of the Doppler frequency, along with the frequency of the laser light and knowledge of the angle at which the two laser beams cross. Namely,

$$V = f_D \frac{\lambda}{2\sin\kappa},$$

where f_D is the Doppler frequency, λ is the wavelength of the laser light, and κ is the half-angle of the beam crossing. It is this equation that we will used to determine the uncertainty in the velocity measurement.

As outlined in Holman (1984), the uncertainty in a calculation based on measured quantities can be calculated by following a general procedure. Consider a quantity R which is a function of n measured quantities, x_1, x_2, x_3, …, x_n, with associated uncertainties w_1, w_2, w_3, …, w_n. We wish to know the uncertainty in the calculated value R, namely w_R. This will be given by:

$$w_R = \left[\left(\frac{\partial R}{\partial x_1} w_1 \right)^2 + \left(\frac{\partial R}{\partial x_2} w_2 \right)^2 + \ldots + \left(\frac{\partial R}{\partial x_n} w_n \right)^2 \right]^{1/2}$$

For the case at hand, we will take the wavelength of the laser light, λ, to be known exactly. For a He-Ne laser, the wavelength is 632.8 nm. For the uncertainty in κ, we must consider the beam spacing and the focal length of the focusing lens. TSI provides an alignment mask for use with their systems. This allows the beams to be positioned accurately

to within ½ of a beam diameter, or 0.5 mm. This, coupled with the 250 mm focal length of the focusing lens, means that the true value of κ is known to within 0.1°, or 2.5% of κ.

The uncertainty in the measurement of the Doppler frequency arises from the accuracy of the TSI counters used in this investigation. The counters were determined by Wilder (1992) to have an accuracy in their measurement of frequency of 0.01*f*, worst case, for the range of frequencies where the Doppler signal is expected to lie. Therefore, the uncertainty in the measurement of the Doppler frequency is $w_{f_D} = 0.01 f_D$.

We can now estimate the uncertainty in the velocity measurement, w_V as,

$$w_v = \left[\left(\frac{\partial V}{\partial f_D} w_{f_D} \right)^2 + \left(\frac{\partial V}{\partial \kappa} w_\kappa \right)^2 \right]^{1/2} = \left[\left(\frac{\lambda}{2 \sin \kappa} w_{f_D} \right)^2 + \left(\frac{- f_D \lambda \cos \kappa}{2 \sin^2 \kappa} w_\kappa \right)^2 \right]^{1/2}.$$

Recalling the expression for V and that $w_{f_D} = 0.01 f_D$ and $w_\kappa = 0.025\kappa$, it follows that,

$$w_v = \left[\left(\frac{V}{f_D} 0.01 f_D \right)^2 + \left(\frac{V}{\tan \kappa} 0.025 \kappa \right)^2 \right]^{1/2} = 0.02689 V .$$

Therefore, the uncertainty in the velocity measurement is $w_V = 0.027$ V.

2.5 Edwards – Datametrics Barocell Precision Pressure Transducer

A Barocell precision pressure transducer was used as a standard to measure the pressures during the calibration of the pressure transducers of the ESP pressure scanner. The ESP pressure scanner will be discussed in the next section. The Barocell model 590D-100T-3Q8-H5X-4D was used in conjunction with a model 1450 Electronic Manometer, which provided an LED readout of the pressure and access to nulling and calibration controls. The Barocell has a range of 100 torr and a resolution of 0.01% of full scale. The stated accuracy of the Barocell is 0.05% of the reading + 0.001% of the full scale.

2.6 The PSI ESP Pressure Scanner

The ESP line of pressure scanners consists of a number of independent miniature silicon piezoresistive pressure transducers mounted together to a common substrate. These pressure transducers are multiplexed to a single amplified output stage with a range of ±5 volts. The transducers are packaged in modules, with the number of transducers going in powers of 2 starting with 8 (8-channels, 16-channels, 32-channels, …). The ESP is essentially a miniature scanning valve without any moving parts, capable of switching from channel to channel, pressure transducer to pressure transducer, at a rate of 20,000 Hz. This allows for measurements to be made on the different channels almost simultaneously. For example, 16 channels can be sampled in 800 μsec. Which channel is multiplexed to the output amplifier is controlled by a 6-bit binary address sent to the ESP. Aside from the range of this binary address, every ESP is functionally identical to the rest of the supporting electronics, i.e. the computer-to-ESP interface and the data acquisition sub-system, allowing different ESP's to be used without requiring any other changes in the hardware setup. The small size of the ESP package allows the ESP to be mounted directly on the model, or in some cases, in the model. This greatly reduces the tubing length required to get from the pressure taps of the model to the ESP, maximizing the frequency response possible for the measurements.

Two ESP pressure scanners were utilized in the current research, a 32-channel model with a range of ±20" of H_20 and a 32-channel model with a range of ±10" of H_20. When properly calibrated, the ESP has an accuracy of 0.10% of the full scale and an acceleration response of ±0.008% of full scale per G. The ESP was calibrated once an hour when data was being taken to insure the integrity of the data. Each of the pressure transducers in the ESP are internally connected to the same reference pressure, which is connected to the REF port on the ESP. For the current research, the reference pressure is the static pressure of the wind tunnel being used. A photograph of an ESP-32 can be seen in Figure 2.7.

2.6.1 ESP Calibration

The pressure transducers of the ESP were calibrated following the guidelines of the manufacturer, Pressure Systems Incorporated (PSI). PSI recommends the following model equation for the ESP:

$$P = C_1 + C_2 V + C_3 V^2,$$

where P is the pressure, V is the measured output voltage of the ESP, and C_1, C_2, and C_3 are constants of the calibration. When calibrated in this fashion, PSI claims an accuracy for the ESP of 0.10% of the full scale of the transducers. To facilitate using this model equation,

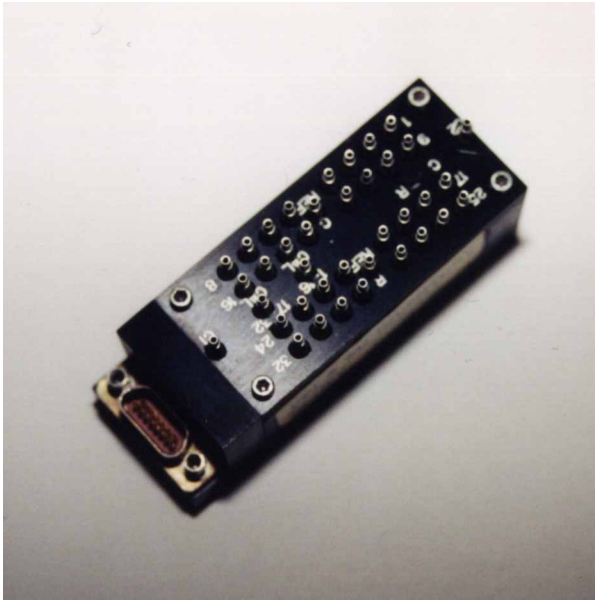


Figure 2.7: An ESP-32 Scanning Electronic Pressure Transducer as used in this investigation. The ESP-32 measures 1"x2.5"x1.8".

three pressures were generated, measured with a standard pressure transducer and with each of the transducers of the ESP. The voltages were then measured from each of the transducers of the ESP and the calibration coefficients for the model equation were found using a least squares fit to the model equation.

To begin the calibration of the ESP, the ESP must be placed in calibration mode. This is an arrangement of the ESP where all the pressure transducers in the package are internally connected to a common manifold. This manifold is connected to a port on the ESP package marked CAL. To get the ESP into calibration mode, 80 psi of dry air is applied to port C1 on the ESP. A pressure is generated using a water tower arrangement, shown schematically in Figure 2.8. An open-ended chamber is mounted to a threaded rod. This chamber is submerged in a container of similar diameter, which is filled with water. The pressure of the air trapped in the changes as the chamber is submerged deeper into the container. By connecting flexible tubing to the chamber, an adjustable pressure generator is created. The pressure generated is measured by the Barocell pressure transducer, which is discussed in Section 2.5. This pressure and the voltages from the ESP pressure transducers are recorded. Usually, the same pressure is then applied to the REF port of the ESP and the voltages recorded again. This makes two of the calibration points, P and –P, where P is the pressure generated by the water tower. The third point of the calibration is the zero offset, where both the CAL port and the REF port of the ESP are exposed to the same pressure, resulting in zero differential pressure across the transducer. With all three of these pressures and voltages recorded, calibration coefficients for each of the transducers in the ESP can be calculated. To switch the ESP back to run mode, where each of the transducers is connected to their corresponding port on the face of the ESP, 80 psi of dry air is applied to the C2 port of the ESP.

2.6.2 Uncertainties in the Measurement of the Coefficient of Pressure

The ESP will be used to measure the pressure, relative to the static pressure of the wind tunnel being used, at each of the pressure taps on the surface of the model. These measured surface pressures will be used in the calculation of the coefficient of pressure (Cp) for that location on the model. In this section, an estimate of the uncertainty in that measurement will be presented. The coefficient of pressure is defined as:

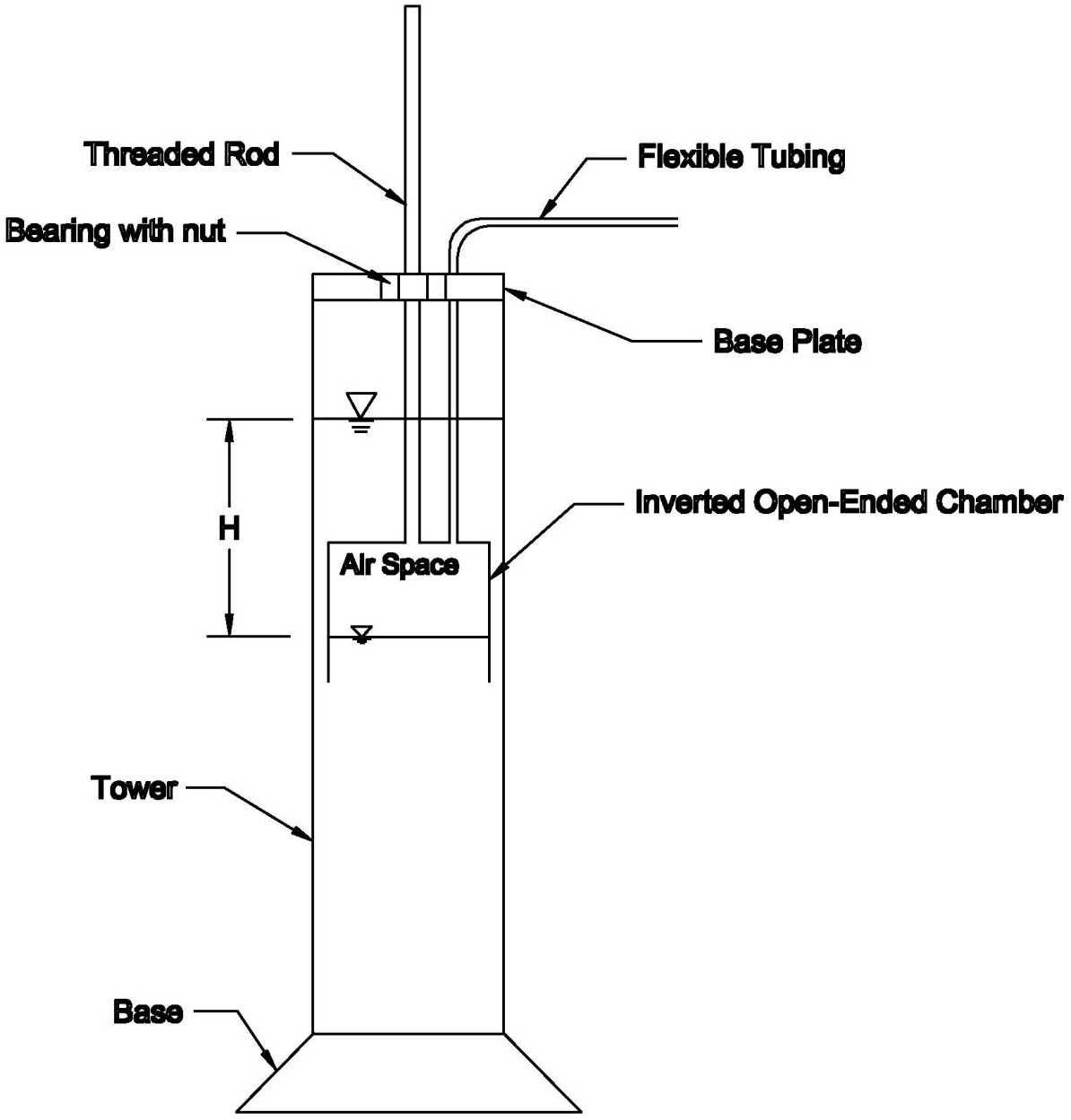


Figure 2.8: Schematic of the Calibration Pressure Generator (Water Tower) (Cross-Section).

$$C_P = \frac{P - P_\infty}{\frac{1}{2}\rho U_\infty^2} = \frac{\Delta P}{P_D}$$

where ΔP is the measured pressure, P, relative to the static pressure of the tunnel, P_∞, and $P_D = \frac{1}{2}\rho U^2_\infty$ is the dynamic pressure. Both of these quantities are measured by the ESP and both are subject to the uncertainties of the ESP.

The uncertainty in a measurement made by the ESP arise from four quantities, namely, the static error of the ESP, the acceleration induced errors of the ESP, errors associated with the analog-to-digital conversion, and the accuracy of the Barocell, since the ESP is calibrated against the Barocell. As noted earlier, the static error of the ESP is 0.1% of the full scale, or ±0.0746 torr for an ESP with a range of ±20" of water and ±0.03731 torr for an ESP with a range of ±10" of water. The acceleration error of the ESP is 0.008% of the full scale/G. The worst case for acceleration of any of the motions considered here is the 20° to 50° ramp in 0.5 seconds, or 1.0472 radians/second. This results in an acceleration error of ±0.0007 torr for the ±20" of water ESP and ±0.0003 torr for the ±10" of water ESP. The accuracy of the Barocell factors into the error of the ESP since the Barocell is used to measure the calibration pressures. The Barocell has an accuracy of 0.05% of the reading plus 0.001% of the full scale. A typical calibration pressure was 6 torr, therefore the uncertainty in the Barocell reading was ±0.004 torr.

Determining the errors introduced by the analog-to-digital conversion involves assuming "typical" values for the calibration coefficients of the ESP. Table 2.1 lists a set of calibration coefficients for both the ±20" of water ESP and the ±10" of water ESP.

ESP	C_1 (torr)	C_2 (torr/volt)	C_3 (torr/volt^2)
±20" of Water	-1.12565	8.40304	0.01706
±10" of Water	0.16815	4.33067	0.00621

Table 2.1: Typical Calibration Coefficients for the ±10" of water ESP and the ±20" of water ESP.

The analog-to-digital conversion was done utilizing a A/D converter with 12-bit accuracy. With a ±5 volt range, this corresponds to a voltage uncertainty of ±0.00122 volts. Using the typical calibration coefficients, this voltage error corresponds to a pressure error of ±0.00528 torr for the ±10" of water ESP and ±0.01025 torr for the ±20" of water ESP. Taking the static error, the acceleration error and the A/D error in quadrature, the total uncertainty in the pressure measurement made by an ESP is ±0.0793 torr for the ±20" of water ESP and ±0.0397 torr for the ±10" of water ESP.

As outlined in Holman (1984), the uncertainty in a calculation based on measured quantities can be calculated by following a general procedure. Consider a quantity R which is a function of n measured quantities, x_1, x_2, x_3, …, x_n, with associated uncertainties w_1, w_2, w_3, …, w_n. We wish to know the uncertainty in the calculated value R, namely w_R. This will be given by:

$$w_R = \left[\left(\frac{\partial R}{\partial x_1} w_1 \right)^2 + \left(\frac{\partial R}{\partial x_2} w_2 \right)^2 + \ldots + \left(\frac{\partial R}{\partial x_n} w_n \right)^2 \right]^{1/2}$$

Therefore for the coefficient of pressure,

$$w_{C_P} = \left[\left(\frac{\partial C_P}{\partial \Delta P} w_{\Delta P} \right)^2 + \left(\frac{\partial C_P}{\partial P_D} w_{P_D} \right)^2 \right]^{1/2}$$

or

$$w_{C_P} = \left[\left(\frac{1}{P_D} w_{\Delta P} \right)^2 + \left(\frac{-\Delta P}{P_D^2} w_{P_D} \right)^2 \right]^{1/2}$$

$$w_{C_P} = \left[\frac{w_{\Delta P}^2}{P_D^2} + \frac{\Delta P^2 w_{P_D}^2}{P_D^4} \right]^{1/2}$$

Since the dynamic pressure and the surface pressure are measured by the same instrument, the uncertainties are the same for these two measurements, i.e., $w_{\Delta P} = w_{P_D} = w$. Therefore,

$$w_{C_p} = \left[\frac{w^2 P_D^2 + \Delta P^2 w^2}{P_D^4} \right]^{1/2}$$

The largest C_p that will be encountered in this investigation is –3.0, accordingly

$$C_P = -3.0 = \frac{\Delta P}{P_D} \Rightarrow \Delta P = -3.0 P_D$$

and

$$\Delta P^2 = 9.0 P_D{}^2 .$$

Assembling all of this, we have

$$w_{C_p} = \left[\frac{w^2 (P_D^2 + 9P_D^2)}{P_D^2} \right]^{1/2} = \left[\frac{10 w^2}{P_D^2} \right]^{1/2}$$

Since the highest dynamic pressure we will encounter is 1.29" of water, or 2.40624 torr, the uncertainty in the measurement of the coefficient of pressure, for the results presented herein, is $C_p \pm 0.1042$ for the ±20" of water ESP and $C_p \pm 0.0521$ for the ±10" of water ESP.

2.6.3 Effect of Flexible Tubing on the Pressure Measurements

Since the pressure transducers are connected to the pressure taps on the model by runs of flexible plastic tubing, the effect of those tubing runs on the measured pressures needs to be considered. All the tubing runs in the present investigation were made of Tygon type 3603-Laboratory flexible plastic tubing with an inside diameter of 0.03125-inch. All the tubing runs were kept less than 12 inches in length. Rediniotis (1992) used a white noise pressure signal with a bandwidth of 200 Hz to show that, for this type and length of tubing, the attenuation and phase lag are insignificant for frequencies up to 25 Hz. This is a tubing frequency response an order of magnitude greater than that of the frequency of the dynamic motions presented herein. For tubing runs of 6 inches, the amplitude response and the phase lag at 25 Hz was found to be 1.00675 and –2.2647°, respectively. For a 12 inch run, the amplitude response and the phase lag at 25 Hz was found to be 1.01768 and –6.3384°, respectively.

2.4 Models

The data presented herein was taken using five different models. All were geometrically similar to one another. A schematic of the basic geometric form of the models is presented in Figure 2.9. All the models were 75° sweep, aspect ratio 1.07 delta wings with a thickness to chord ratio of 0.0496 and a leading-edge bevel of 45°. To aid in identifying what model was used in what phase of the investigation, the following names will be used. Two 0.14-meter (5.598-inch) chord models of aluminum construction were used in all the water tunnel work. One was equipped with cavity flaps and the other was not. These will be referred to as the LDV models with or without flaps. A 0.28-meter (11.2-inch) chord model,

constructed with an hollow aluminum body and a steel deck, was used in the ESM Wind Tunnel. This model was equipped with a set of deployable cavity flaps and will be referred to

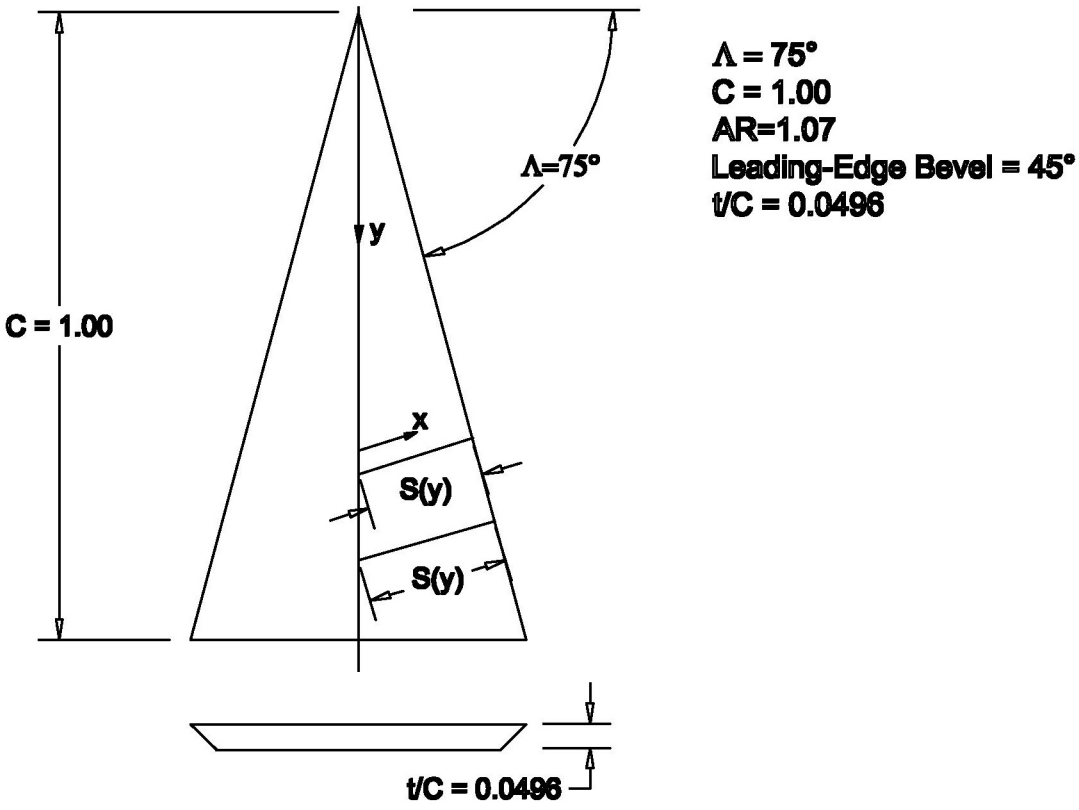


Figure 2.9: The basic geometric layout for all the models. All the models, the LDV, Red, Black, and White are all geometrically similar to this schematic.

as the Red model. Two models were used in the Virginia Tech Stability Wind Tunnel, a model of aluminum construction with a 0.66-meter chord and one of aluminum and steel construction with a chord length of 1.0-meter. The 0.66-meter chord model was equipped with a deployable apex flap and will be referred to as the White model. The 1.0-meter chord model was tested, at various times, without any flaps, with stationary cavity flaps, and with

deployable cavity flaps. It will be referred to as the Black model. With the cast of characters and the scene set, it is time to proceed with the story.

CHAPTER 3: THE THREE-DIMENSIONAL FLOW FIELD OVER A DELTA WING EXPERIENCING VORTEX BREAKDOWN

Of all the tools a worker in fluid mechanics has at their disposal, flow visualization is one of the most basic and often turned to tools. Most experimental studies of flow fields start with some type of flow visualization. For example, smoke, bubble and dye injection are used to study streamline patterns and oil flows are utilized to study the surface streamline patterns. Traditionally, these have been tools in the arsenal of the experimentalist. However, the advancements in computational fluid mechanics has led some to develop routines for taking the computed velocity field from simulations and mimicking the experimental flow visualizations to allow comparisons of the computed results with those from experiments. Furthermore, these techniques can be enhanced to accomplish what is impossible in a laboratory situation.

Interestingly, the experimentalist also finds a need for these new tools. The availability of laser Doppler Velocimetry and automated data acquisition has made it possible to map out complex three dimensional flows in very high detail. It is now almost routine to measure all three components of velocity on a very tight grid. In short, it is now possible to generate a tremendous amount of data that needs to be managed and reduced into meaningful information. So, experimentalists find themselves turning to tools originally developed for workers in computational fluid dynamics to assist them in the digestion of their data and to identify flow structures hidden within.

Although these tools were originally conceived to mimic the flow visualizations done in the laboratory, they can be made to be superior to the original. Items can be coded by color, or have colors mapped on to them that relate to the variation of a different variable. The ability to cut three-dimensional objects to examine their internal structure is another item not readily available in the laboratory. Moreover, it is important, critical in three-

dimensional flows, to develop techniques that give the researcher a feeling for "how the flow goes", an insight into the physics of how the flow is moving through space.

In this chapter, extensive three-component LDV measurements will be presented. This data will allow the entire velocity and vorticity field of a leading-edge vortex to be reconstructed, permitting the study of the topology of the sectional streamlines (the streamlines constructed from the in-plane velocity vectors), the stability of the foci within those planes, and the distribution of vorticity on planes before and after the breakdown location. In addition, we extend the field of view to the three-dimensional streamline structures and to the three dimensional distribution of normalized helicity and axial velocity. The concept of feeding zones will be introduced.

Thus far in the literature, with the exception of Rockwell and his associates, only the axial component of vorticity has been presented. The behavior of the other components of vorticity, such as the azimuthal component, which is very indicative of breakdown, has not been explored. Also, very little experimental information on the topologic structure of leading-edge vortex breakdown has been presented to date. The results presented here offer an opportunity to compare such information to their computationaly-derived counterparts.

3.1 Experimental Conditions

The experiments were conducted in the Engineering Science and Mechanics (ESM) Water Tunnel, using the TSI LDV system discussed in Chapter 2. All the data presented within this chapter was taken with the LDV model without flaps (0.144-meter chord). The model was mounted within the tunnel at an angle of attack of 38°. This angle of attack was chosen so that vortex breakdown occurs about mid-chord. The Reynolds number, based on chord length, was 72,000. Measurements were taken on 22 planes normal to the free-stream. These planes were uniformly spaced with a spacing of $\Delta X/C' = 0.028$, where $C' = C \cos \alpha$ is the projection of the wing chord length along the free-stream. The first and last measurement planes were positioned at $X/C' = 0.348$ and 1.046, respectfully, with $X/C' = 0$

being the apex. Note that this coordinate system differs from the others used herein. Here X is distance down the centerline of the wing, while elsewhere the coordinate used is y. There is data for about 65% of the wing length. All three components of velocity were acquired at a total of about 12,500 points. From the three components of velocity, the three components of vorticity were calculated. The data has not been smoothed in any way. A schematic of the measuring system is shown in Figure 3.1.

As setup this portion of the investigation the LDV system, with its laser mounted along side the system instead of underneath, as typically is for this LDV system, was mounted upon an optical bench and positioned so that the optical train is normal to the axis of the tunnel. Directly in front of the optical train, a mirror tower was used to bring the beams into the focusing optics and also to traverse the measuring volume. Two mirror towers were used depending upon if the data was being taken from the side or the top of the tunnel. Figure 3.2 shows the relationship between the measured velocity components and the test section. When data is being taken from the side, the measured velocity components are along the x' and z' axes, or equivalently along x and z, as shown in Figure 3.2(a). Two mirrors are used to bring the beams through the focusing optics and into the test section. Moving the two mirrors as a unit facilitates traversing in the y direction. Moving the two mirrors relative to one another, accomplishes the z traversing. When data is taken from the top, the measured velocity components are along the x" and y" axes, or equivalently, along x and y as shown in

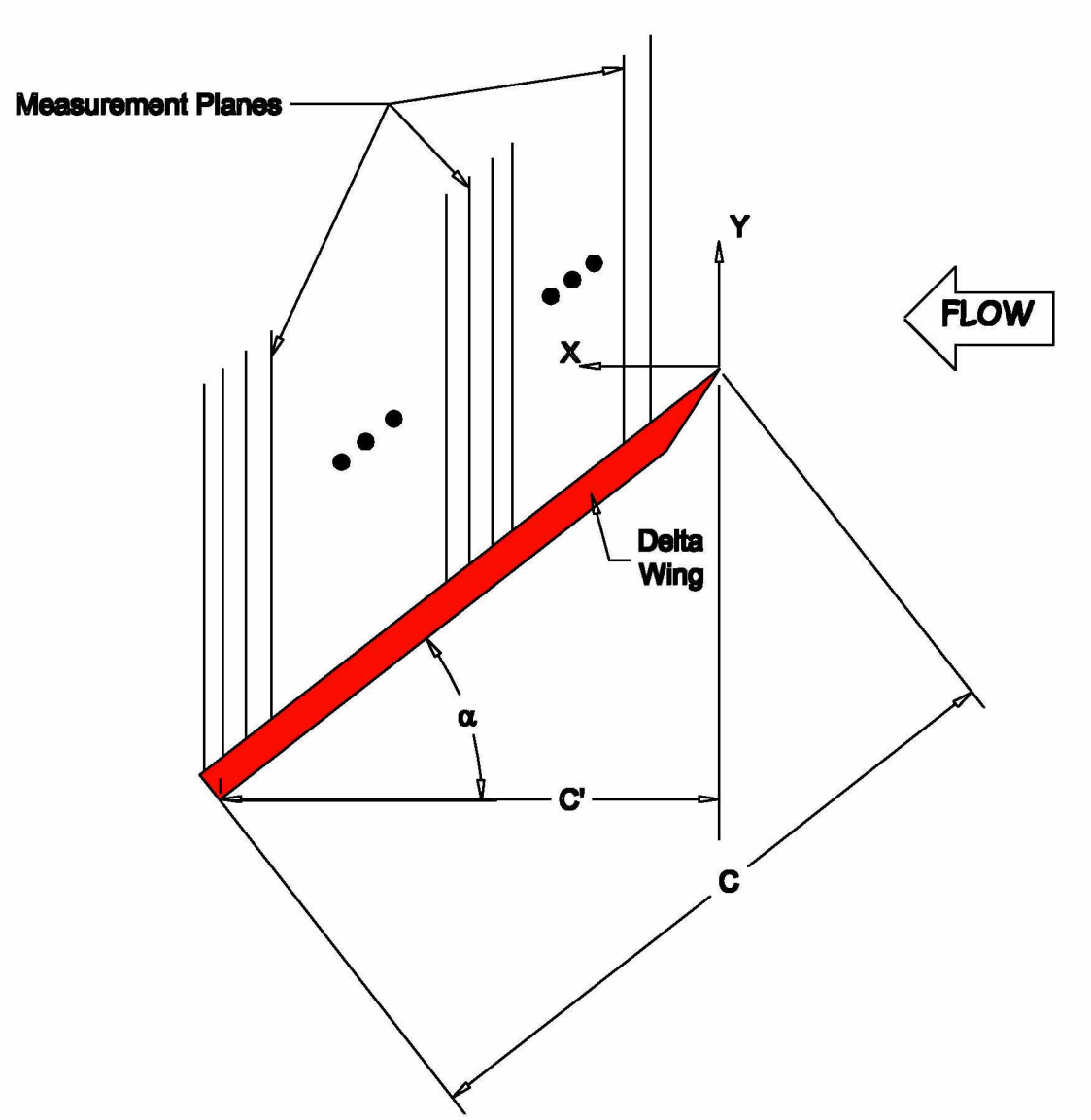


Figure 3.1: Schematic of the Measurement Domain and the Coordinate System Definition.

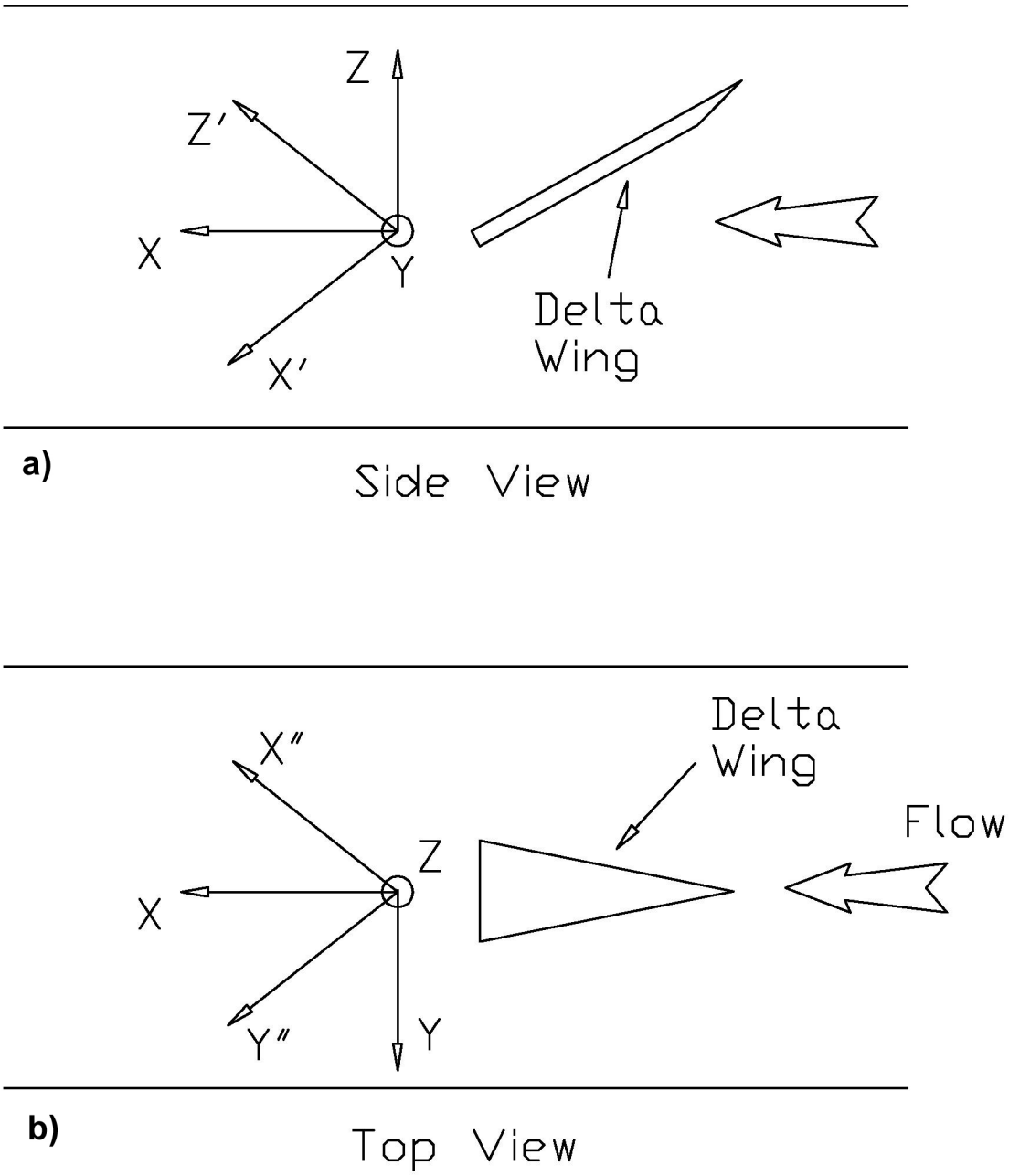


Figure 3.2: The 3-D LDV setup and measurement axes, a) the arrangement when taking data from the side, and b) the arrangement when taking data from the top.

Fig. 3.2b. Three mirrors are required to bring the beams through the focusing optics and into the test section. Moving all three mirrors as a unit accomplishes the y traverse, while moving the top two relative to the bottom one provides movement of the measuring volume in z. In each case, a 250 mm f/4 lens focuses the beams into a measuring volume. This arrangement allows the same plane, which is normal to the flow, to be traversed from the top and from the side therefore allowing all three components of the velocity be obtained, although not all simultaneously.

Traversing in the y and z directions is automated by two stepping motors and monitored by two LVDT's. The latter provide an independent analog feedback to confirm the accurate positioning of the measuring volume. Two-dimensional grids in planes parallel to the y-z plane can be traversed. The accuracy in positioning the measuring volume is 0.2mm. To obtain data along different y-z planes, the entire optica1 bench is mounted upon a manual traversing system and can be traversed in the x-direction. The entire data acquisition process is fully automated. The only processes requiring manual operation are traversing in the x-direction (for a different plane of measurement) and seeding the water about every 12 hours.

The visualization of the flow field was accomplished using AVS, a commercially available scientific visualization package produced by Advanced Visual Systems, Incorporated. One of the greatest strengths of AVS is its flexibly. The user defines the visualization process by building a network of subroutines for that particular project. The building of the network is very similar to visually building an object-oriented program. However, in most cases, no code has to be written. The user simply picks the modules that are needed, e.g. to read data from a file, construct a vector from scalar data, or render a vector field. Data flows from one module to another until the data has been rendered on the screen. At this point it is possible for the user to pan, rotate the image in three-dimensional space, move slicing planes and other objects, or zoom in or out, all interactively with the mouse. AVS was run and all of the

images rendered on a Silicon Graphics 4D/430-VGX Power Series with 64 Mbytes of RAM. These facilities are part of the Virginia Tech Laboratory for Scientific Visual Analysis.

3.2 Results

Since the leading edge vortex is of primary importance to delta wing aerodynamics, it makes sense to begin by looking at quantities that are basic to vortex flows. Figure 3.3 presents two views of the leading edge vortex. The data was acquired only on the starboard side of the model, but for this figure the data was mirrored to the port side to allow two views of the vortex simultaneously. The gray object is a rendering of the wing itself. On the port side we see an isosurface of vorticity magnitude. The isosurface is set at a non-dimensional vorticity magnitude, defined as $|\Omega|c/U_\infty$, of 33.45. On the starboard side, streamlines are shown in the core region of the vortex. These streamlines are calculated by using the fact that for a steady flow, streamlines and pathlines are the same. By allowing massless particles to convect through the flowfield, computationally, we can plot the pathlines and hence the streamlines. All of the streamlines presented here were calculated in this fashion. Also shown in Figure 3.3 is an isosurface that correspond to the region of the flow with zero axial velocity. The existence of a region of zero axial velocity is indicative of the vortex breakdown. The streamlines have a color map applied to them based on the local value of the axial velocity using a standard “rainbow” or spectrum color map. Red in the color map corresponds to high axial velocity and the blue to low axial velocity. It can be seen that the flow in the core of the vortex is decelerating well before the breakdown location indicated by the isosurface of zero axial velocity. The streamlines are yellow for a large distance upstream. Then they shift through green to blue in the region of the breakdown bubble. The “radius” of the streamline bundle also increases in front of and as they pass over the bubble. The vorticity magnitude isosurface is seen to stop at the same station down the wing that the zero axial velocity surface begins. The high vorticity of the organized core is destroyed by breakdown. This station is at an X/C' = 0.744.

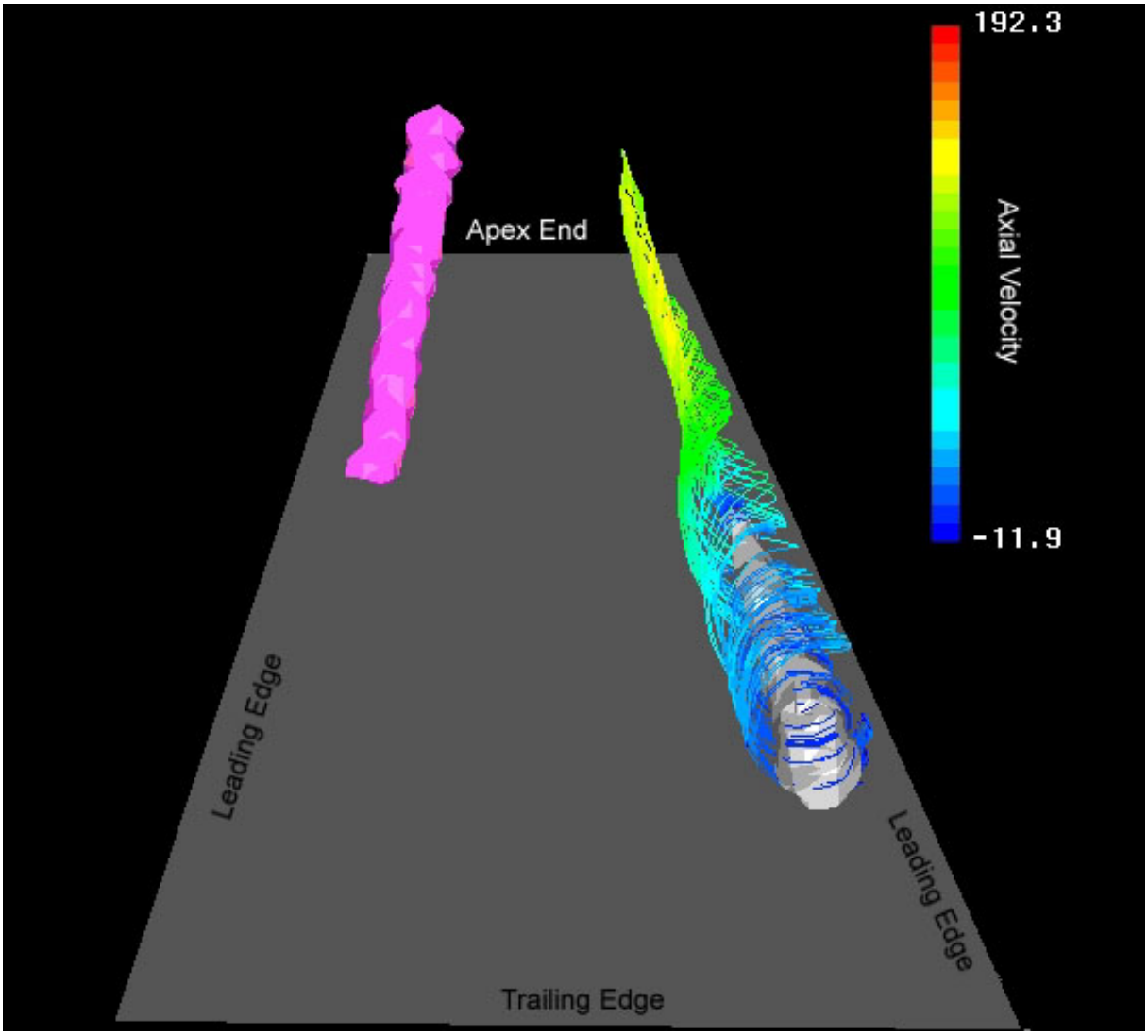


Figure 3.3: Isosurface of vorticity magnitude (pink) and zero-axial velocity (white). Streamlines presented are color-mapped based on the local value of the axial velocity. The data has been mirrored about the wing centerline.

The source for all the vorticity in the leading edge vortex system is the separating shear layer that originates at the leading edge. If we look at the streamlines that emanate from the leading edge, as shown in Figure 3.4, we see that instead of rolling uniformly into the vortex, there appear to be axial locations that are preferred. The streamlines for this case were advanced using a Runge-Kutta technique with step size 0.002 for 128 time steps. The starting points for all the streamlines are in uniform locations along the leading edge. These

preferred locations are actually the start of a secondary vortex system. This vortex system holds its orientation with respect to the primary vortex. This is why the structure shows up in time averaged data. Squire inferred their existence in 1961(Squire, 1961) on the basis of oil flow patterns. Washburn and Visser (1994) also documented these structures for steady flow over a delta wing at low angles of attack. Their persistence even in the presence of vortex breakdown has not been previously noted. The presence of the structures can be clearly seen by rendering the streamlines as a mesh instead of individual lines. This is seen in Figures 3.5 and 3.6. The point of view of Figure 3.5 is looking down on the planform of the wing. The point of view for Figure 3.6 is looking from the centerline of the wing outboard. It is interesting to note that the streamlines that emanate from the leading edge do not get anywhere near the core before they are swept off the wing. The majority of them complete only one spiral. This means that all the vorticity in the core of the vortex is entrained from the leading edge forward of the first plane of data that we have here, i. e. upstream of X/C' = 0.348. Only a length of the leading edge within the first 34.8% of the leading edge contributes vorticity that makes it to the core.

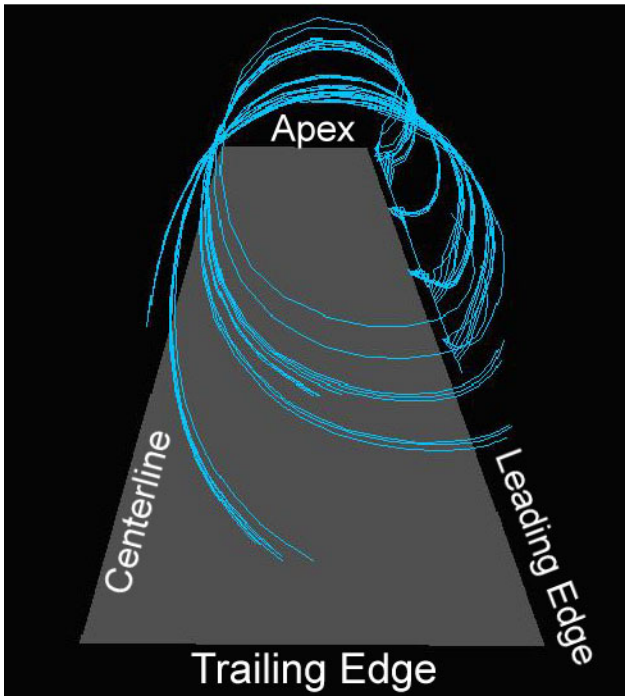


Figure 3.4: Streamlines emanating from the leading-edge of the wing. View is looking upstream from behind the wing.

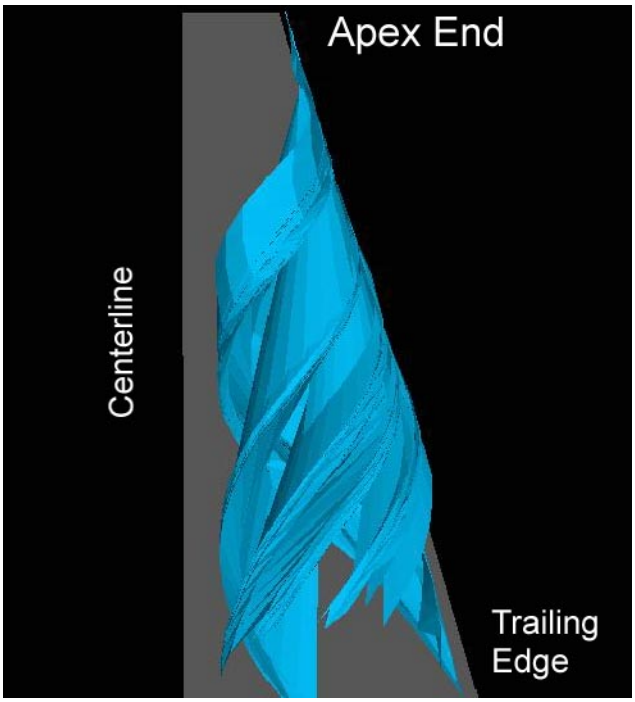


Figure 3.5: Streamline emanating from the leading edge of the wing rendered as a mesh. Looking down on the wing planform from above.

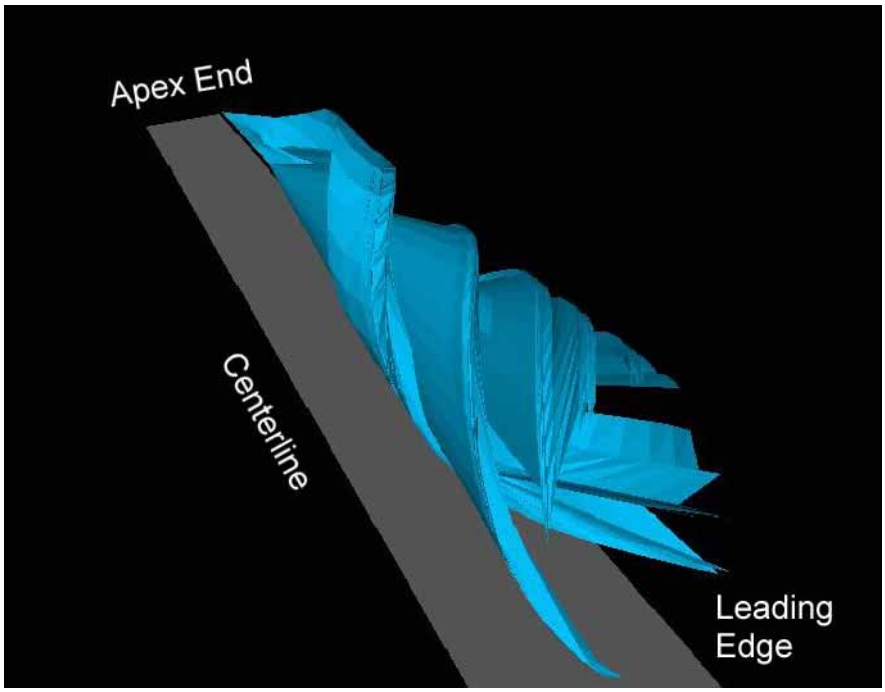


Figure 3.6: Streamlines emanating from the leading edge rendered as a mesh. Looking from the centerline outboard.

Figure 3.7 presents the distribution of all three components of vorticity along several measurement planes. These are planes normal to the free stream at the following x/C′ locations: 0.463, 0.522, 0.579, 0.665, 0.696, 0.723, 0.839. The contours illustrate that distribution of the axial vorticity component. Throughout the entire paper, unless otherwise noted, the term "axial" indicated direction perpendicular to the measurement plane, i.e. the free-stream direction. The other two vorticity components are presented by in-plane vectors. The values of vorticity shown along the legend are non-dimensional. Several points are worth noticing here. The axial velocity distribution for this angle of attack (Rediniotis, 1992), indicates that breakdown occurs at about x/C′=0.655, which is slightly upstream of plane 4 (x/C′ = 0.665). However the axial vorticity distribution at plane 4 does not shown appreciable signs of breakdown. This is not surprising and has already been observed for the case of vortex breakdown over a delta wing in a pitch-up maneuver. The above observation

suggests that of the two quantities, axial velocity or axial vorticity, the best qualified candidate for early sensing and subsequent control and prevention of breakdown in a high-alpha maneuver, is the axial velocity. A very interesting phenomenon, predicted by theory, however not previously documented experimentally as descriptively as here, is the sign reversal of the azimuthal component of vorticity.

In Figure 3.7, planes 1, 2, 3, which are located upstream of breakdown, exhibit a clear counter-clockwise sense (considered positive) of the azimuthal vorticity. This is compatible with Biot-Savart law, which relates the azimuthal vorticity to the axial velocity. Positive azimuthal vorticity induces positive axial velocity to decrease from high positive values upstream of breakdown to zero and finally negative values inside breakdown, the azimuthal vorticity has to undergo a dramatic change in order to account for that deceleration. And so it does by totally reversing its sign. Clear signs of reversal are evident in plane 4. The succession of Figures 3.7 through 8 illustrates the evolution of this reversal. Figure 3.8 presents sectional streamline patterns along two planes, upstream (3.8(a)) and downstream

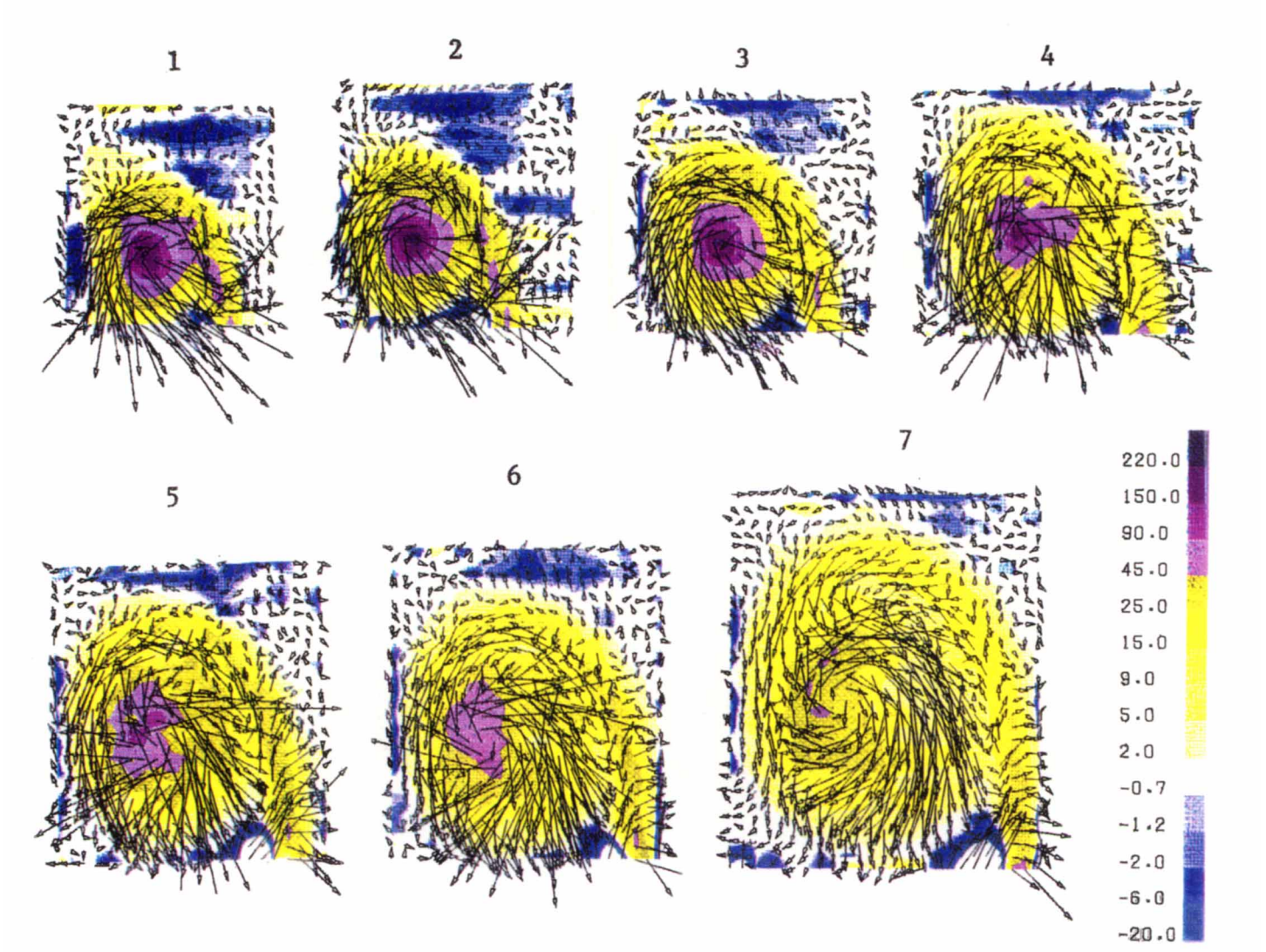


Figure 3.7: The behavior of the three vorticity components along several planes before and after breakdown. The arrows are the in-plane components of vorticity, the colors represent the out-of-plane component.

(3.8(b)) of breakdown. The plane in 3.8(a) is oriented normal to the wing planform. The topology of Figure 3.8(a) is characterized by the following critical points: a half-saddle S'_1 on the centerline of the wing, a half-saddle S'_2 at the leading edge, a saddle point S_3 on the wing's plane of symmetry and an unstable focus on N_1. A few comments should be made here about the nature of focus N_1. The type of a critical point (saddle, node, focus) as well as its nature (attracting/repelling) can be provided by the characteristics of the rate-of-deformation tensor $\partial u_i/\partial x_j$ in the neighborhood of the critical point. We will employ some of this analysis to resolve a question pertaining to the unstable nature of focus N_1, which does not seem to agree with the traditional representation of a leading-edge vortex featuring a stable focus. The trace *Tr* of the 2-D rate-of-deformation tensor (along the cross section under consideration) contains the information to resolve the issue. In the neighborhood of the focus if $Tr < 0$ the focus is stable and if $Tr > 0$ it is unstable (Perry and Chong, 1987). The trace is nothing but the divergence of the velocity: $Tr = \partial v/\partial y + \partial w/\partial z$, with v, w being the velocity components on the cross section and u being the component normal to it. However, since our flow is incompressible, the trace is also equal to $Tr = -\partial u/\partial x$, due to continuity. As we have seen previously (Figure 3.3), the axial velocity u in the core of the vortex upstream of breakdown undergoes deceleration, *i.e.*, $\partial u/\partial x < 0$, therefore $Tr = -\partial u/\partial x > 0$, which in turn means that the focus is in fact unstable. We found that for this angle of attack ($\alpha = 38°$) this is the case for the entire vortex core upstream of breakdown and downstream of $X/C' = 0.35$, Magness et al. (1993) have documented similar behavior for the case of a delta wing in a pitch-up motion. The existence of an unstable focus does not really contradict the traditional stable-focus representation since the latter refers to lower angles of attack where breakdown either does not occur over the wing or has not advanced far upstream. By observing the distribution of $\partial u/\partial x$ on streamlines near the core, we can see how far upstream of the breakdown location, the focus first becomes unstable. In Figure 3.9, streamlines near the core are shown along with the isosurface of zero axial velocity. The streamlines have color mapped to them according to a very simple color map. Values of the trace which are less than

zero, and hence infer stability of the focus, are colored light blue, while values of the trace which are greater than zero, and hence imply an unstable focus, are colored red. We can see that the focus first becomes unstable at an X/C' = 0.592, far upstream of the zero axial velocity isosurface.

Figure 3.8(b) presents sectional streamlines along a plane normal to the wing, downstream of a breakdown. The pattern is characterized by two limit cycles and an unstable focus at the core. The inner limit cycle is an attracting one. Streamlines inside the cycle spiral outwards and collapse onto it, while streamlines outside the cycle spiral inwards towards it. The latter streamlines emanate from the outer limit cycle, which is a repelling one. The pattern was also examined on a cross section normal to the free stream to make sure that it stays unaltered.

Figure 3.10 presents a set of streamlines in the immediate neighborhood of breakdown. Spiral/saddle S_1 is the stagnation point in the upstream face of the breakdown bubble. An expected rear saddle point is probably located further downstream of the domain shown here. It is interesting to see that the breakdown bubble is not closed. Flow from upstream enters the bubble at locations downstream of the saddle S_1, and once it reaches it, it spirals out away from it.

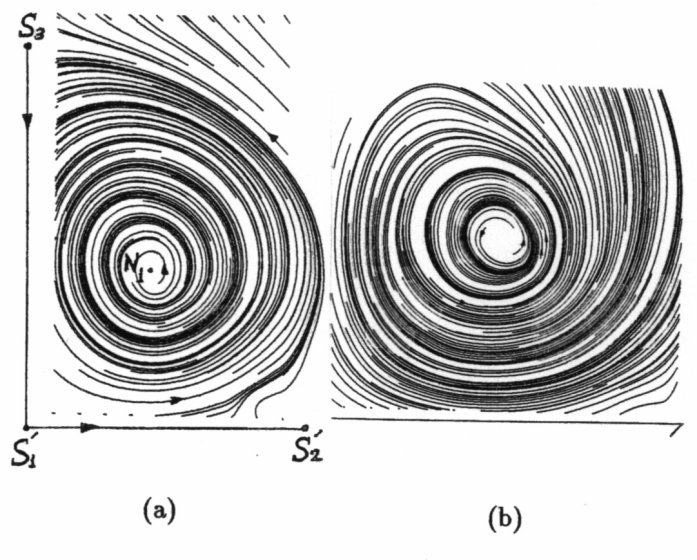


Figure 3.8: Sectional streamline pattern along a plane normal to the wing, (a) upstream of breakdown, (b) downstream of breakdown.

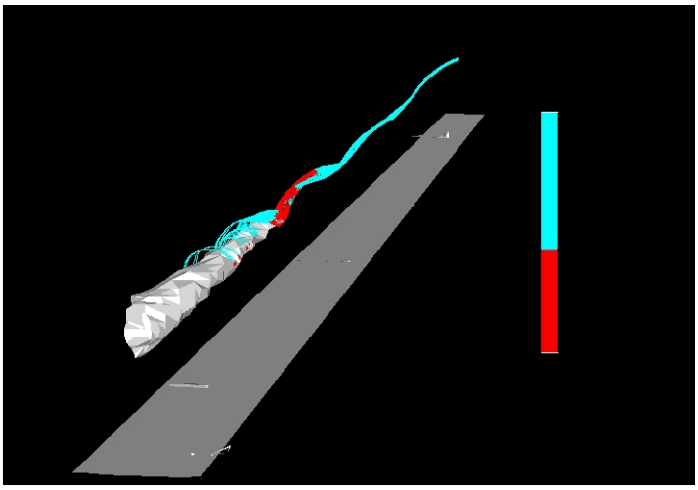


Figure 3.9: Variation of the axial velocity gradient along the core of the vortex. Light blue indicated that the focus is stable, red indicated that the focus is unstable.

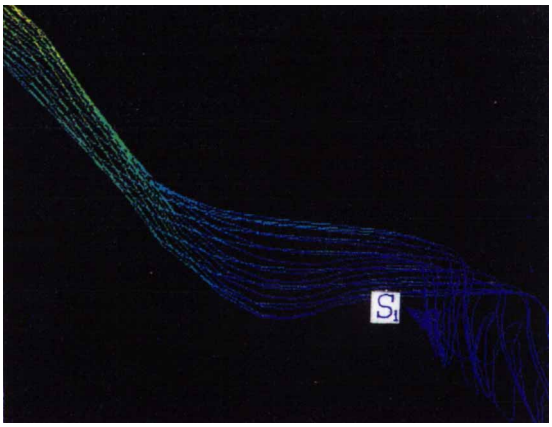


Figure 3.10: Behavior of the three-dimensional streamlines in the neighborhood of breakdown.

It should be mentioned again here, that all the patterns shown here are for time-averaged data and as such represent the time-averaged flow patterns. In a pattern, such as the one shown in Figure 3.8, the two saddle points S'_1 and S_3 would appear to have a common separatrix normal to the wing. However, it is known that saddle-to-saddle connections are unstable and can be easily broken by a perturbation in the flow (Perry and Chong, 1987). Experimental evidence for that is provided in Figure 3.11. A helium-bubble technique was employed to visualize the cross section of the leading-edge vortices. The picture in Figure 3.11 was taken in the VPI Stability Wind Tunnel, with a model geometrically similar to the one employed in the water tunnel, the Black model with a chord length of 1 meter. In this wind tunnel technique, soap bubbles, filled with a mixture of air and helium to achieve neutral buoyancy, are injected into the flow through a hole in the model side near the apex. The bubbles are quickly entrained into the core of the vortex, due to the proximity of the hole to the apex. A light sheet cuts the model in a plane aligned perpendicular to the free-stream. In an otherwise darkened tunnel, only the bubbles in the light sheet are illuminated. The result is the photograph shown in Figure 3.11. The vortex and its core are clearly visible. The bubbles were injected into the flow through a hole on the left side of the model. Therefore they should only be entrained into the left vortex.

However, bubble paths can be seen in the right vortex also and they could only get in there through periodic rapture of the separatrix between saddles S'_1 and S_3.

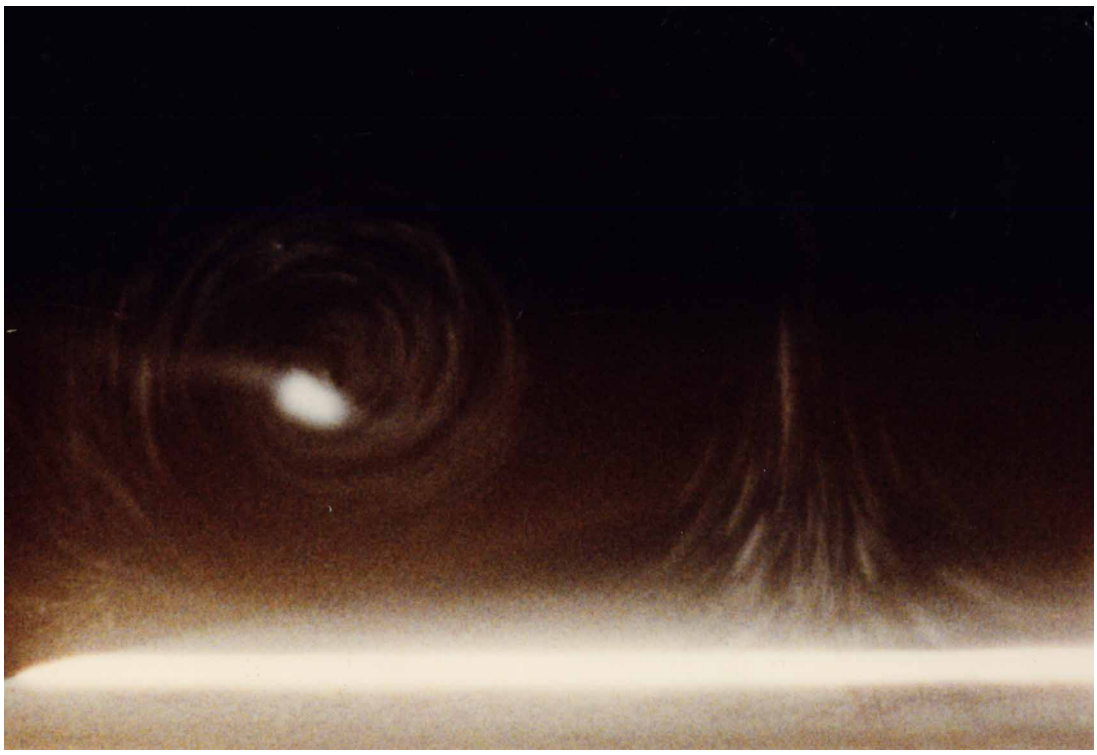


Figure 3.11: Visual evidence of the unstable nature of the saddle-to-saddle connection between the two delta wing vorticies.

The image in Figure 3.11 was the inspiration for the development of a new visualization technique for the leading-edge vortex. It would be informative to look at a sectional cut similar to that in Figure 3.11, but with the particle traces within the “light sheet” color coded as to the origin in the flow of the streamline that the particle trace is part of. By color coding the starting location of each streamline we can identify how fluid particles, or streamlines, which originate at the leading edge or anywhere upstream are incorporated into the structure of the leading edge vortex. Several start sites for the streamlines are selected. By varying the viewing plane, the "light sheet", it can be seen how different parts of

the vortex are "fed" by streamlines from different parts of the flow. Hence the name, feeding zones.

All the of the streamline systems used have their starting points in the first plane of data. This is because, as noted earlier, none of the streamlines emanating from the leading edge within the data volume are able to move into the core region. Four hundred streamline starts were placed on six lines, 25 starts per line. The lines were oriented in the vertical direction at locations: Red L/h = 0.197, Gold L/h = 0.345, Yellow L/h = 0.507, Cyan L/h =0.676, Magenta L/h = 0.831, and Blue L/h =1.0, where L is the distance from the centerline of the wing to the line and h is the distance from the centerline to the leading edge at the first data plane. Assuming the flow is conical, this arrangement should capture the streamlines closest to the core region. The total streamline system can be seen in Figure 3.12. The streamlines were advanced using a Runge-Kutta technique with step size 0.0005 for 95 time steps.

Figure 3.13 is the result of cutting the streamline system at various X/C' planes. In (a), which is at X/C' = 0.522, we see that the yellow streamlines must be in plane with the vortex core, for above the center they move out board, below they move inboard. Following several of the colors, magenta, for instance, shows up spiraling into the core. Even the blue lines, which started from the leading edge, are spiraling in. In (b), the location is shifted downstream to X/C' = 0.697. We already see evidence of the beginning of breakdown. Some of the yellow streamlines have been moved away from the core, indicating again that the core has become an unstable foci instead of a steady foci. This is an earlier indication then what the zero axial velocity isosurface provides. In (b), the axial velocity seems strongest near the centerline of the wing, as evidenced by the longer path lengths of the streamlines within the cutting plane. In (c), breakdown is very evident. Situated at X/C' = 0.871, we are well in the breakdown region. All the streamlines have move away from the vortex center, with the exception of one single yellow line. The axial velocity is very small at this station as

evidenced by only small line segments appearing. Fig 12 (d), is the flow slightly beyond the trailing edge at X/C' = 1.039, and is included for completeness.

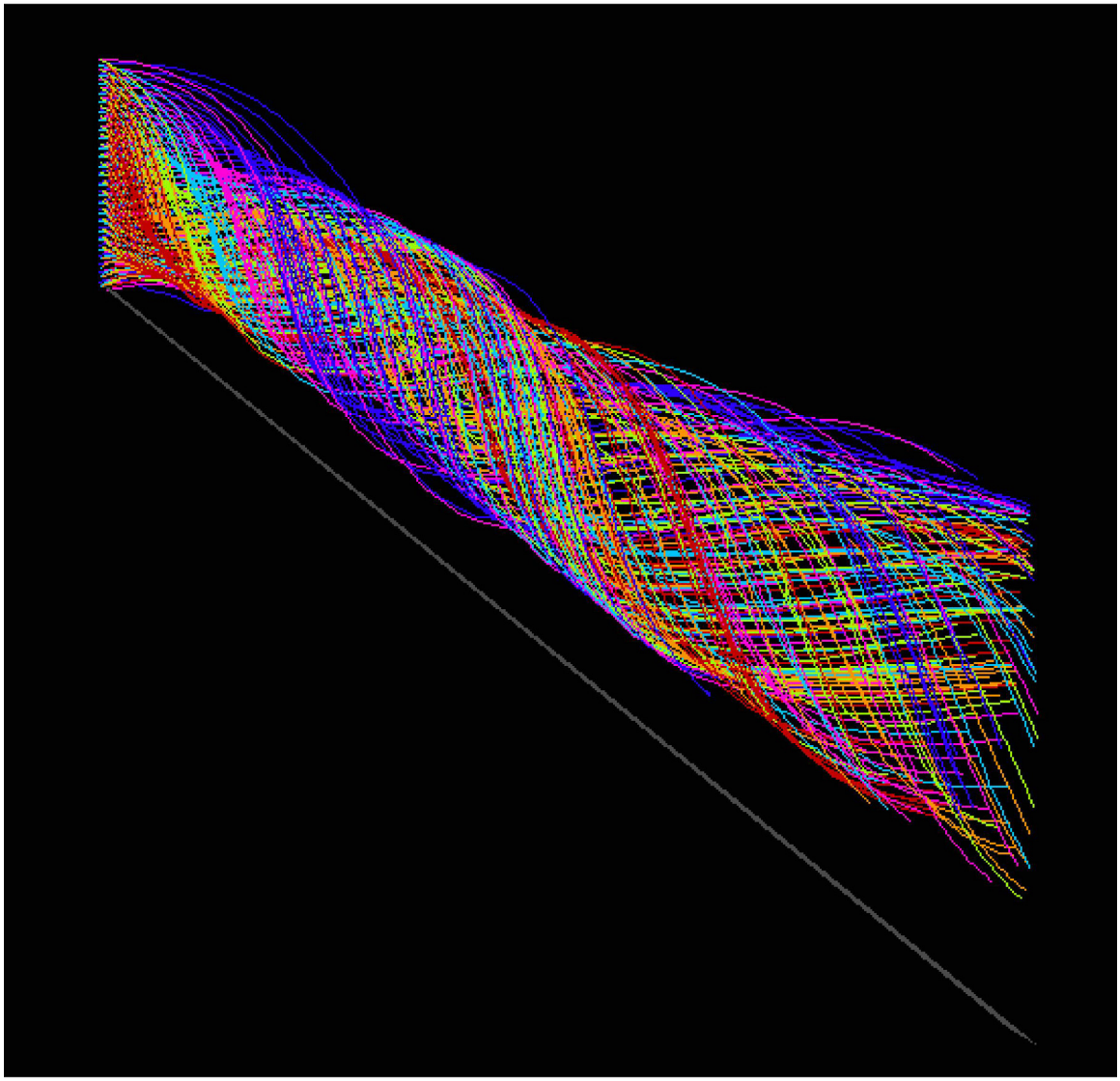


Figure 3.12: All the streamlines used in the feeding zone analysis seen from the centerline of the wing and looking outboard.

Helicity has been applied to the visualization of fluid flows, especially vortical flows, with great success (Levy, et. al., 1990). Moffet (1969) introduced helicity, in its fluid mechanics interpretation, and its ever increasing role was summarized years later by Moffet

and Tsinober (1992). Moffet defined helicity to be an integral quantity, integrated over the entire flow field. He went on to define a value of, what he termed, helicity density, as the dot

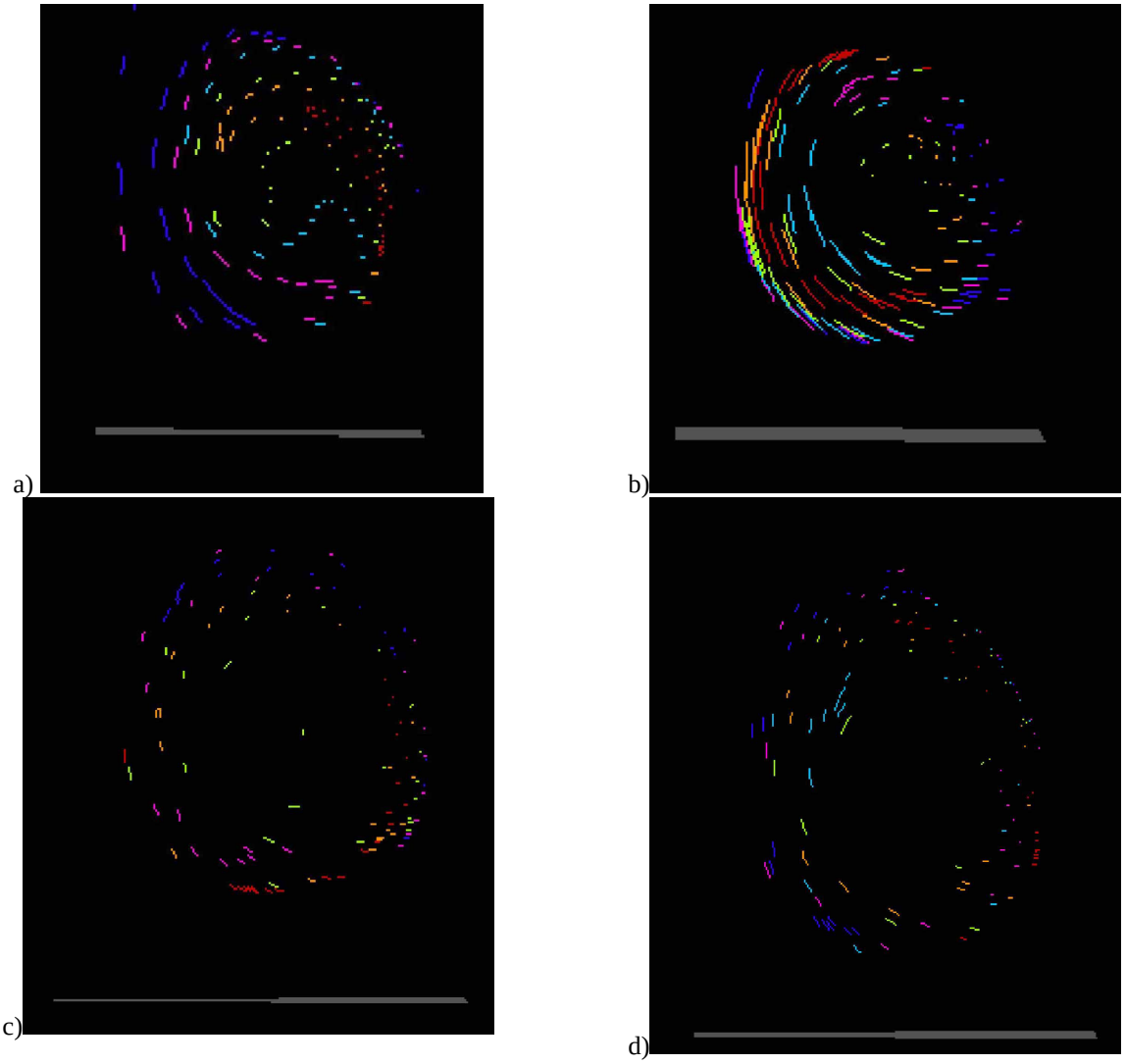


Figure 3.13: Feeding zone planes: (a) x/C=0.522, (b) x/C'=0.679, (c) x/C'=0.781, (d)x/C'=1.0309. Images are not to the same scale.

product of the velocity and vorticity vectors at some point in the flow field or,

$$H_d = V \bullet \omega$$

A somewhat more useful quantity is the so-called normalized helicity (Levich and Tsinober, 1993). Normalized helicity is defined as the helicity density divided by the product of the magnitudes of the velocity and vorticity vectors or,

$$H_n = \frac{V \bullet \omega}{|V||\omega|}$$

Geometrically, the normalized helicity is the angle, actually the cosine of the angle, between the velocity vector and the vorticity vector. Normalized helicity is very well suited to the visual analysis of vortical flows due to the fact that it is a scalar. It encapsulates within a scalar value, a portion of the wealth of information contained in a vector field. This makes its visual representation very straightforward. In addition, both the magnitude and the sign of the normalized helicity are important. The magnitude allows comparison of relative strengths and the sign gives information about the sense of swirl. Values approaching 1.0 indicate that the velocity vector and the vorticity vector are coming into line with on another, since the normalized helicity is just the cosine of the angle between the two. The fact that the vorticity vector and the velocity vector move from being parallel with in the core of the coherent vortex, to being at an angle to one another is seen quite clearly in Figure 3.14. In this figure, the same region of space is visualized as in Figure 3.10, right in the neighborhood of the saddle S_1. It is quite clear the vortex lines are diverging away from the streamlines as breakdown is encountered. This indicates that the flow has more of an inviscid nature upstream of breakdown, as indicated by the fact the vortex lines and the streamlines are nearly coincident. The flow takes on a more viscous character post-breakdown, where the streamlines and the vortex lines diverge.

In Figure 3.15 we present the distribution of normalized helicity in several planes. Figures 3.15 (a) and (b), show the distribution of normalized helicity in planes at X/C' = 0.348 and 0.522, respectively. We see in each a well organized and coherent vortex. At X/C' =

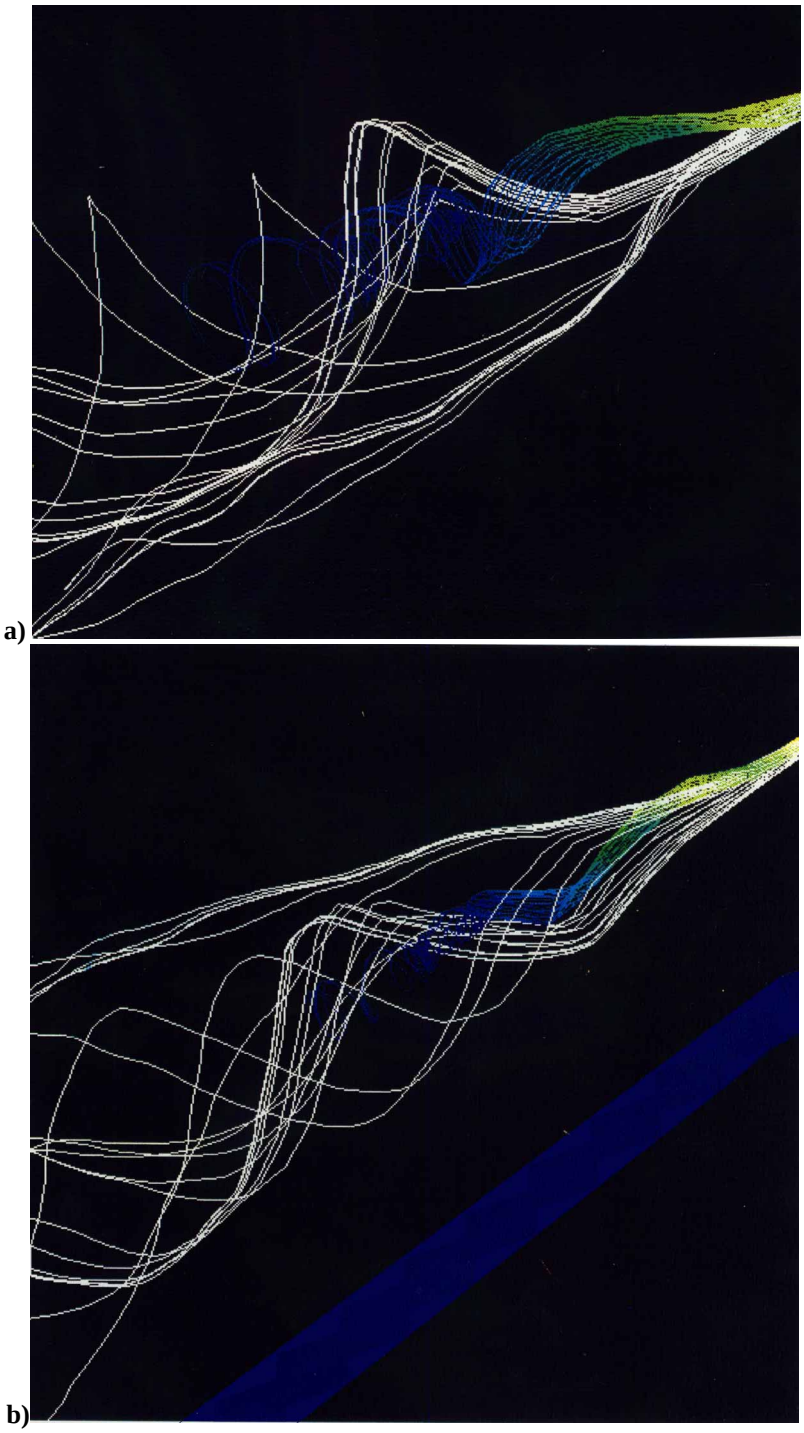


Figure 3.14: Streamlines and vortex lines in the neighborhood of breakdown.

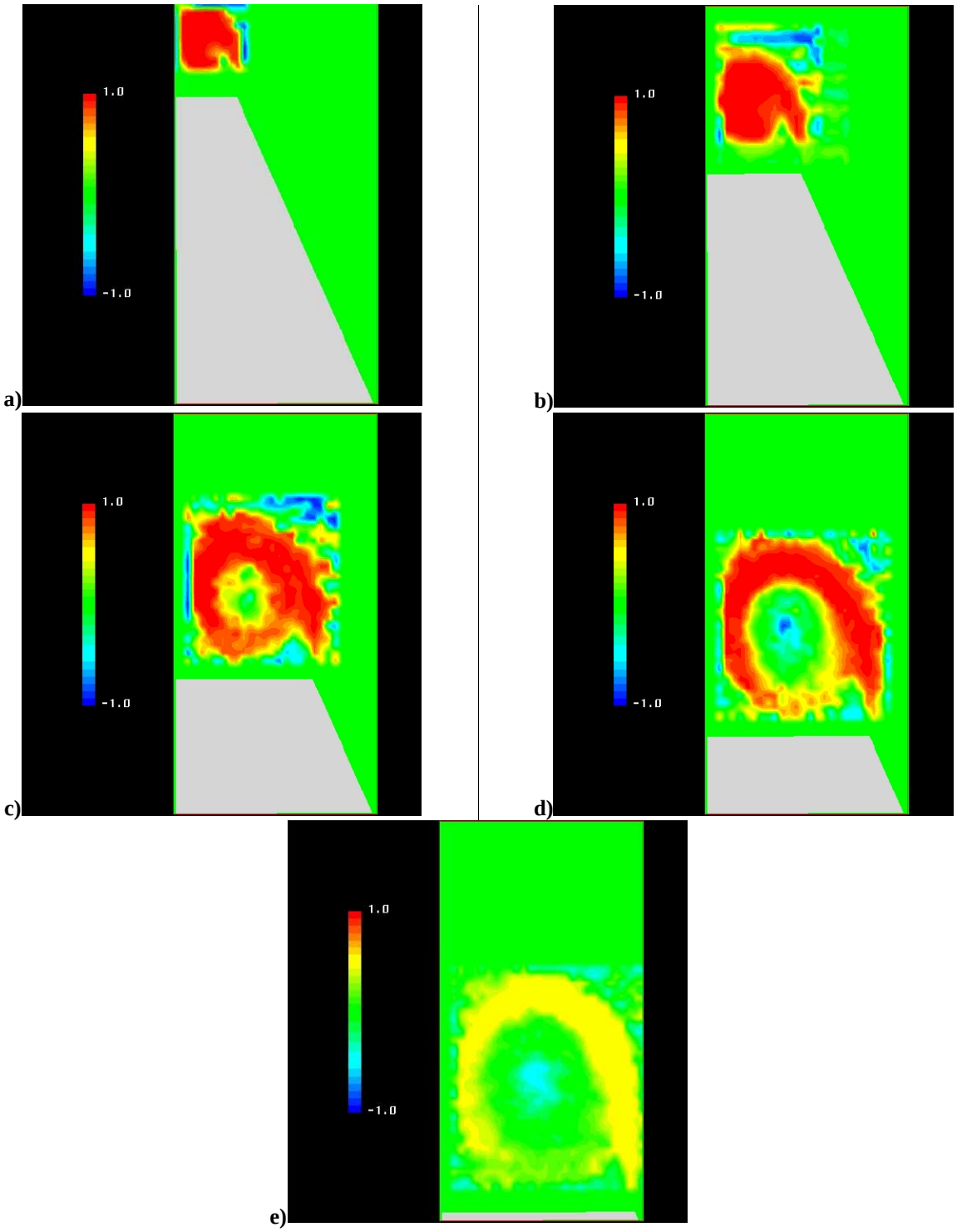


Figure 3.15: Normalized helicity planes: (a) X/C'=0.348, (b) X/C'=0.522, (c) X/C'=0.742, (d) X/C'=0.871, (e) X/C'=1.029

0.742, which is shown in Figure 3.15(c), the helicity shows evidence of the impending breakdown. Two structures can be seen as the lowest value of helicity in the core region and the lower portion of the vortex is also becoming disorganized. In Figure 3.15(d), there exist a region of negative helicity. This plane is located at X/C' = 0.871. This is to be expected for there is significant backflow within the breakdown bubble. Recall that it was shown that streamlines from upstream enter the breakdown bubble and propagate upstream within the bubble. The negative region of helicity persists to the trailing edge as seen in Figure 3.15(e), which is at X/C' = 1.029. The vortex overall has lower values of helicity than before.

CHAPTER 4: THE EFFECT OF CAVITY FLAPS

In this chapter, we explore the potential of employing cavity flaps to control the aerodynamic characteristics of the leading edge vortices and therefore the stability characteristics of delta wings at high angles of attack. A first step in this direction for angles of attack as high as 50 degrees was reported by Rao (1987) who demonstrated that the thrust and drag influence of flaps remain effective well beyond the stall regime. Schematics of a delta wing equipped with cavity flaps are presented in Figure 4.1. This configuration is attractive for fundamental work because it does not change the nominal area of the wing and therefore any changes that occur are due only to the deployment of the flap and not to increase in wing planform. The cavity flap configuration is also attractive from an airframe designer's point of view. They could be applied to either sharp or blunt leading-edges and are structurally superior, as an attachment to the wing, to traditional leading–edge flaps. It is expected that the cavity flap will help modify and control the onset, growth and shedding characteristics of large-scale vortices.

Most of the work presented in the literature on flows over flapped delta wings has been based on force and moment measurements. At best, one can therefore only offer conjectures as to what exactly happens in the development of vortical structures over swept lifting surfaces. Herein, flow visualization, velocity data, and surface pressure measurements for the flow over a delta wing with and without the cavity flaps will be presented, for steady and unsteady flow. This will allow the effects of the flaps to be seen directly.

As mentioned in Chapter 1, this research effort was carried out in three different tunnels of varying turbulent intensities, ranging from 0.03% for the Stability Tunnel to 1.0% and 1.5% for the ESM Water Tunnel and the ESM Wind Tunnel, respectively. It is reasonable to ask if there will be an effect of the level of free-stream turbulence on the occurrence of vortex breakdown. This question does not have a definitive answer. Delery (1994) remarks that "in many circumstances turbulence does not play an essential role in the

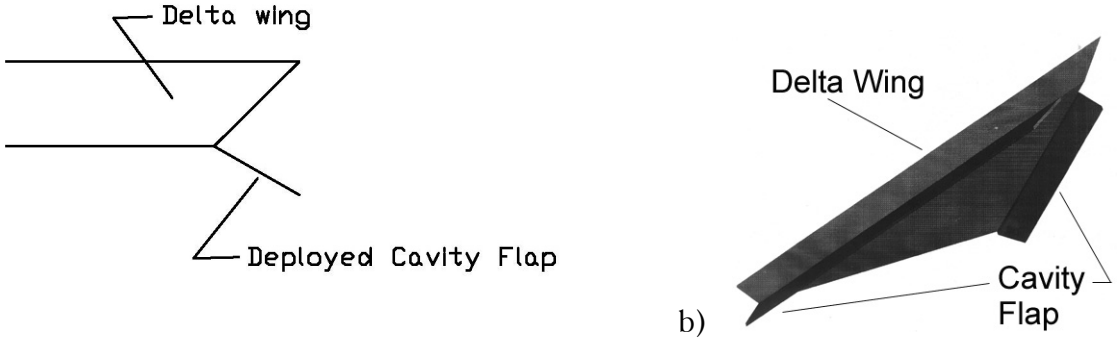
breakdown itself". Sarpkaya (1995) flatly rebuts that statement. Sarpkaya goes on to state that the Reynolds number and the turbulence level play an important role in determining what type of breakdown will occur. The conical shape being the form normally seen at high Reynolds number. Agrawal et. al. (1992) compared the breakdown location for a 70°-sweep delta wing at a Re=1,000,000 the lower portion of the vortex is also becoming disorganized. In Figure 3.15(d), there exist a region of negative helicity. This plane is located at X/C' = 0.871. This is to be expected for using an Euler solution, a laminar solution and a turbulence-model solution. They find that the laminar solution and the turbulent solution give nearly the same results, with the laminar solution agreeing better with the experimental data than the turbulent solution. Given that the location of vortex breakdown over a given configuration is accepted as being Reynolds number independent, the effect of the turbulence levels will only effect the post-breakdown characteristics, e.g. what type of breakdown it is, rather than the breakdown location. This will allow the results from one facility to be compared to the results from another.

4.1 The Performance of Cavity Flaps in Steady Flow

4.1.1 Experimental Conditions

The surface pressure measurements and flow visualizations presented in this chapter were done in the Virginia Tech Stability Wind Tunnel, which was described in Chapter 2, using the Black model (1-meter chord). In some cases, the model was equipped with a set of stationary cavity flaps constructed of sheet steel. These were attached to the bottom surface of the model using the screws that held the bottom in place. The model was tested with and without these flaps in place. Flow visualizations of the flow over the model were accomplished by introducing helium bubbles into the flow through a small hole in the model near the apex. A Model 5 Multi-Head Bubble Generator by Sage Action, Incorporated was used to create nearly-neutrally buoyant helium-filled soap bubbles. These bubbles are very

fine and are pushed through a length of flexible tubing by compressed air. The bubbles were illuminated by a bright light source and photographed using 1600 speed 35 mm film.



a) b)

Figure 4.1: Schematics of the Cavity Flaps Configuration, a) A cross-sectional view through the wing, b) A three-dimensional rendering of a cavity flap equipped delta wing.

Surface pressure measurements were in the Virginia Tech Stability Tunnel made using the ±20-inch of water ESP, via the ESPIO-1 interface. Twenty-seven pressure taps were installed along an axial line on the upper surface of the wing, as shows in Figure 4.2. The taps have an opening diameter of 0.033" and a spacing of 0.019, when non-dimensionalized by the chord of the model. The ESP is connected to the pressure taps by tubing of no more than 12 inches in length. Roughly equal lengths of tubing were used to connect each pressure tap to the ESP. Surface pressure measurements were taken for angles of attack from 28° to 44° in 2° increments.

The LDV measurements of the axial velocity were done in the ESM Water Tunnel using the LDV model with flaps, shown schematically in Figure 4.3. The model was equipped with a set of cavity flaps constructed of modeling plastic and epoxied to the bottom of the model. A photograph of the model is presented as Figure 4.4. The TSI, three-beam, two-component, single color LDV system, described in Chapter 2, was used with the

following traversing scheme. The LDV system, with its laser mounted along side the system instead of underneath, is mounted upon an optical bench and positioned so that the optical train is normal to the axis of the tunnel. Directly in front of the optical train, a mirror tower is used to bring the beams into the focusing optics and also to traverse the measuring volume. Figure 4.5 shows the relationship between the measured velocity components and the test section. Data was being taken from the side only, in contrast to the data outlined in Chapter 3, which was taken from both the side and the top of the tunnel. The measured velocity components are along the x' and z' axes, or equivalently along x and z, as shown in Figure l. Two mirrors are used to bring the beams through the focusing optics and into the test section. Moving the two mirrors as a unit facilitates traversing in the y direction. Moving the two mirrors relative to one another accomplishes the z traversing. A 250-mm f/4 lens focuses the beams into a measuring volume.

Traversing in the y and z directions is automated by two stepping motors and monitored by two LVDT's. The latter provide an independent analog feedback to confirm the accurate positioning of the measuring volume. Two-dimensional grids in lines parallel to the y-axis and the z-axis can be traversed. The accuracy in positioning the measuring volume is 0.2mm. To obtain data along different y-z planes, the entire optica1 bench is mounted upon a manual traversing system and can be traversed in the x-direction. The entire data acquisition process is fully automated. The only processes requiring manual operation are traversing in the x-direction (for a different plane of measurement) and seeding the water about every 12 hours.

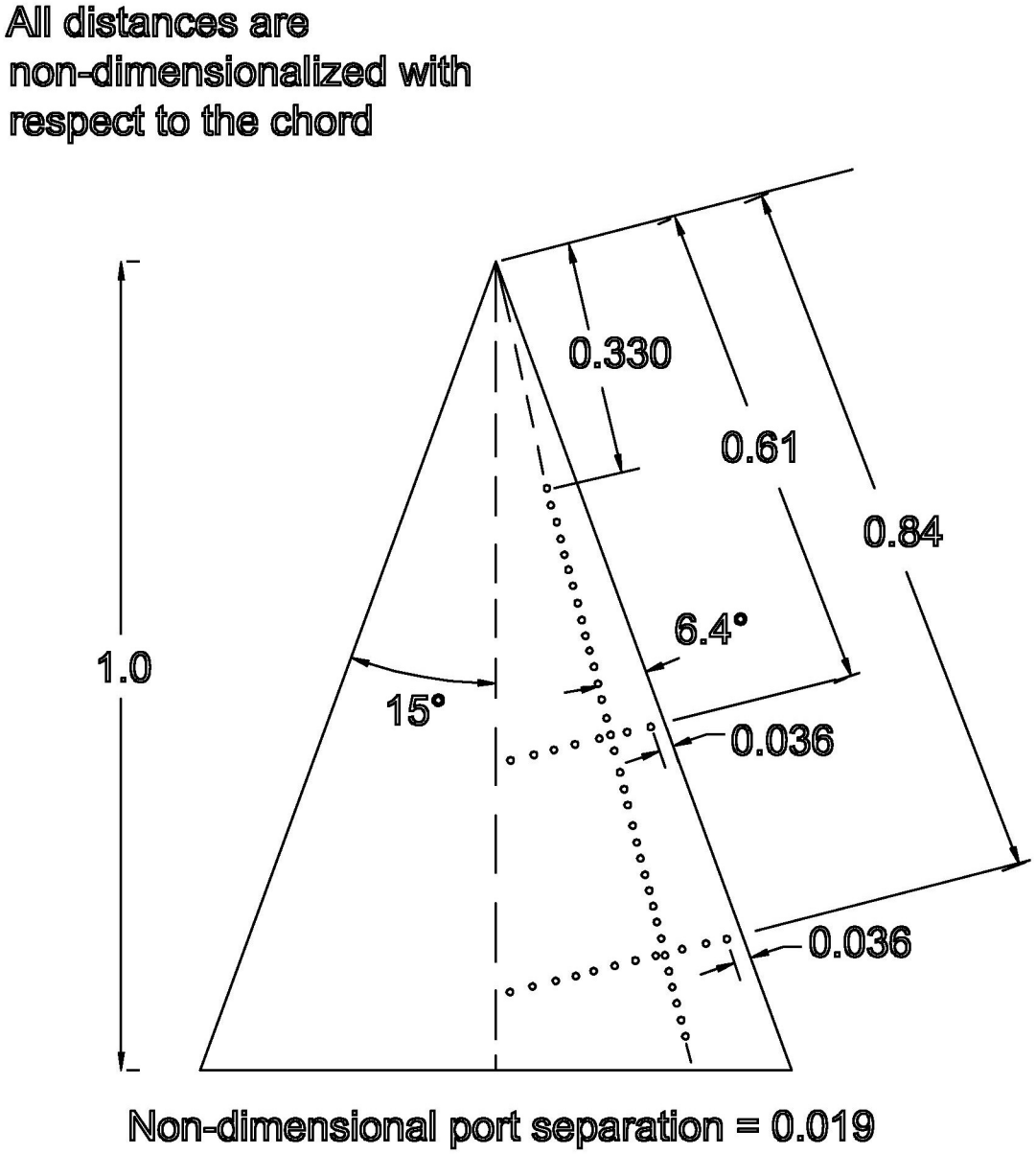


Figure 4.2: Schematic of the Black model showing pressure tap locations for the axial and the spanwise direction.

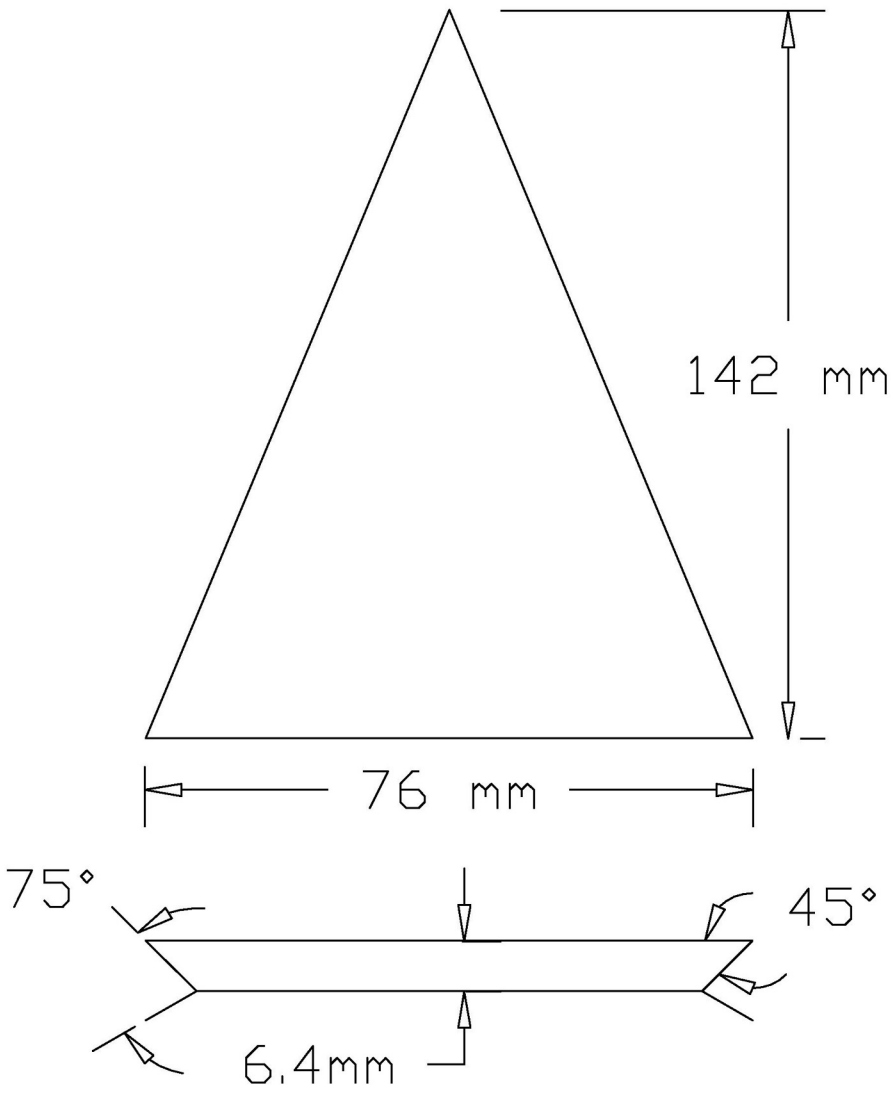


Figure 4.3: Schematic of the LDV Model with Cavity Flaps.

For the LDV measurements, the model was placed in the water tunnel at an angle of attach $\alpha = 35°$. Measurements were obtained along three vertical planes normal to the stream and located at distances y/c = 1.0, 0.75, and 0.50, where y is the distance down the centerline of the model measured from the apex and c is the chord length of the model, as shown in Figure 2.8. These planes will be referred to as planes A, B and C, respectively, as

shown in Figure 4.5. In each plane, a two-dimensional grid was defined with spacing $\Delta y = \Delta z = 1.25$ mm.

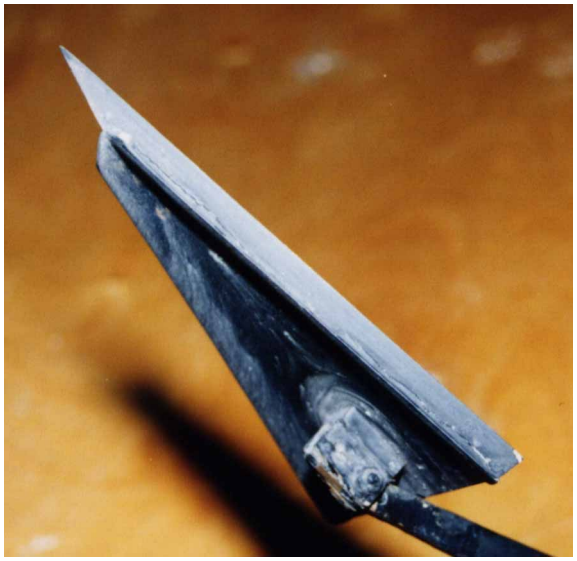


Figure 4.4: Photograph of the LDV model with cavity flaps. The sting used to hold the model in the water tunnel is visible in this photograph.

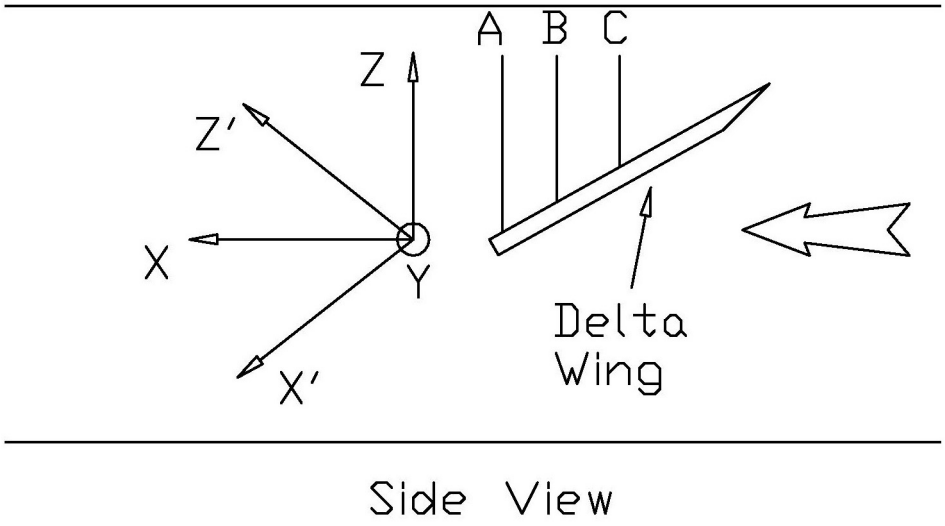


Figure 4.5: Schematic of the LDV setup showing coordinate system used and location of the planes on which the axial velocity measurements were made.

4.1.2 Flow Visualization

As stated earlier, flow visualizations were achieved by releasing helium-filled soap bubbles from a hole near the apex of the wing. At the edge of the wing, the bubbles find themselves inside the vortex sheet and most of them are move quickly to the core of the vortex, where the pressure is at a minimum. The core of the vortex is thus clearly visualized in the form of a line populated densely by bubbles. At breakdown, the core opens up and the region of the flow seeded with bubbles expands.

In Figures 4.6 through 4.9, flow visualizations are presented for angles of attack α = 22°, 26°, 28°, 30°, 32°, 34°, 36° and 40° respectively. In each figure, the top frame shows the visualized vortex core for the plain wing, while the bottom frame shows the results obtained over a wing with deployed cavity flaps. Figure 4.6 presents the flow visualizations for α=22° and α=26°. There are no visible effects of the cavity flaps on the path of the vortex core. In Figure 4.6c, an interesting effect can be noted. The core of the vortex, after passing over the trailing edge of the wing, aligns itself within a short distance with the free-stream. It is this response that is a precursor to what we will see as the angle of attack increases. The corner through which the core of the vortex must turn becomes increasingly sharper as the angle of attack increases resulting in pressure gradient along the core that becomes more and more adverse. This is perhaps the root cause of vortex breakdown. In Figure 4.7a and 4.7b, we observe that the core vortex is coherent all the way to the trailing edge of the wing, although at that location, the vortex over the plain wing is broken down. By comparing Figure 4.7c to 4.7d, it can be seen that vortex breakdown is displaced downstream. In Figure 4.7c, the plain delta wing, vortex breakdown appears at approximately y/C=0.65, where y is the distance down the wing measured from the apex and C is the chord length. While in Figure 4.7d, the cavity flap equipped wing, the breakdown location is at y/C=0.81. Also, the rate at which the vortex core expands upon breaking down is less for the case where the cavity flaps are deployed. It was noted by Payne and Nelson (1987) that "While the bubble type of breakdown has been observed in low Reynolds number water tunnel studies, it is the spiral

type which is more commonly observed in wind tunnel studies." The wind tunnel Reynolds number they used in their investigation was 85,000 based on chord length. Here, the Reynolds number is 470,000, based on chord length. Observations during the experiment and also the photographs presented here indicate that the breakdowns at this Reynolds number should be classified as conical. It seems that while the global structure of the leading-edge vortex is Reynolds number independent, the local features are not.

In Figure 4.8a, the vortex is broken down at y/C=0.48 of the way down the chord, while in Figure 4.8b, the cavity flaps have moved breakdown back to a y/C=0.69. Similar effects can be seen at α=34° (Figure 4.8c and 4.8d) and α=36° (Figure 4.9a and 4.9b). Figure 4.9c presents the flow over the plain delta wing at angle of attack of 40°. The vortex is completely broken down all the way to the apex. In Figure 4.9d, the flow over the cavity flap equipped model at an angle of attack of 40°, there appears what looks like a "double breakdown". The core of the vortex has two expansions with the second, appearing at y/C=0.61, being much more dramatic than the first. It is not known exactly what causes this to occur.

4.1.3 Axial Velocity Distribution

The axial component of the velocity is a strong indication of the location of leading edge vortices. This is because the vorticity wrapping around the vortex sheet induces an axial component of the velocity, which on many occasions can reach values as high as three times the free-stream velocity. Axial velocity contours for a plane wing are plotted in Figure 4.10a, 4.10b and 4.10c along planes A, B and C, respectively, as defined in Figure 4.5. The Reynolds number is 75,000. Figure 4.10c presents evidence of a coherent vortex with a maximum axial velocity $u/U_\infty \approx 2.5$ at a y/C of 0.50. However, the contours of Fig. 4.10b, y/C=0.75, indicate clearly that the vortex core has broken down. The velocity decreases towards the center of the core and it is possible that a stagnated region exists. The pattern is asymmetric and in the outboard region a pocket of high axial velocity exists. This

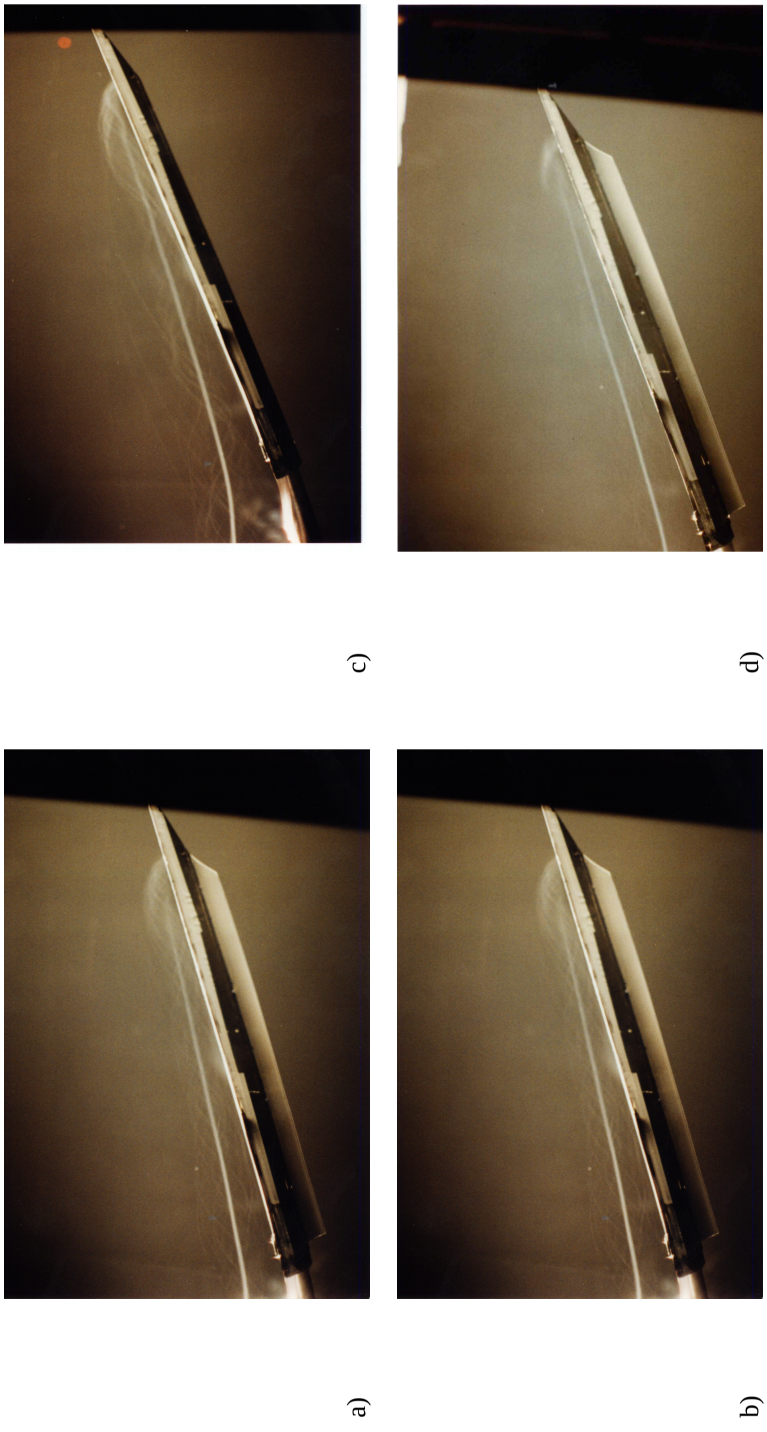


Figure 4.6: Flow Visualization of the effects of cavity flaps. a) No flaps, α=22°, b) Cavity flaps installed, α=22°, c) no flpas, α=26°, d) flaps installed, α=26°. Model: Black (1-meter chord) Tunnel: Virginia Tech Stability Wind Tunnel.

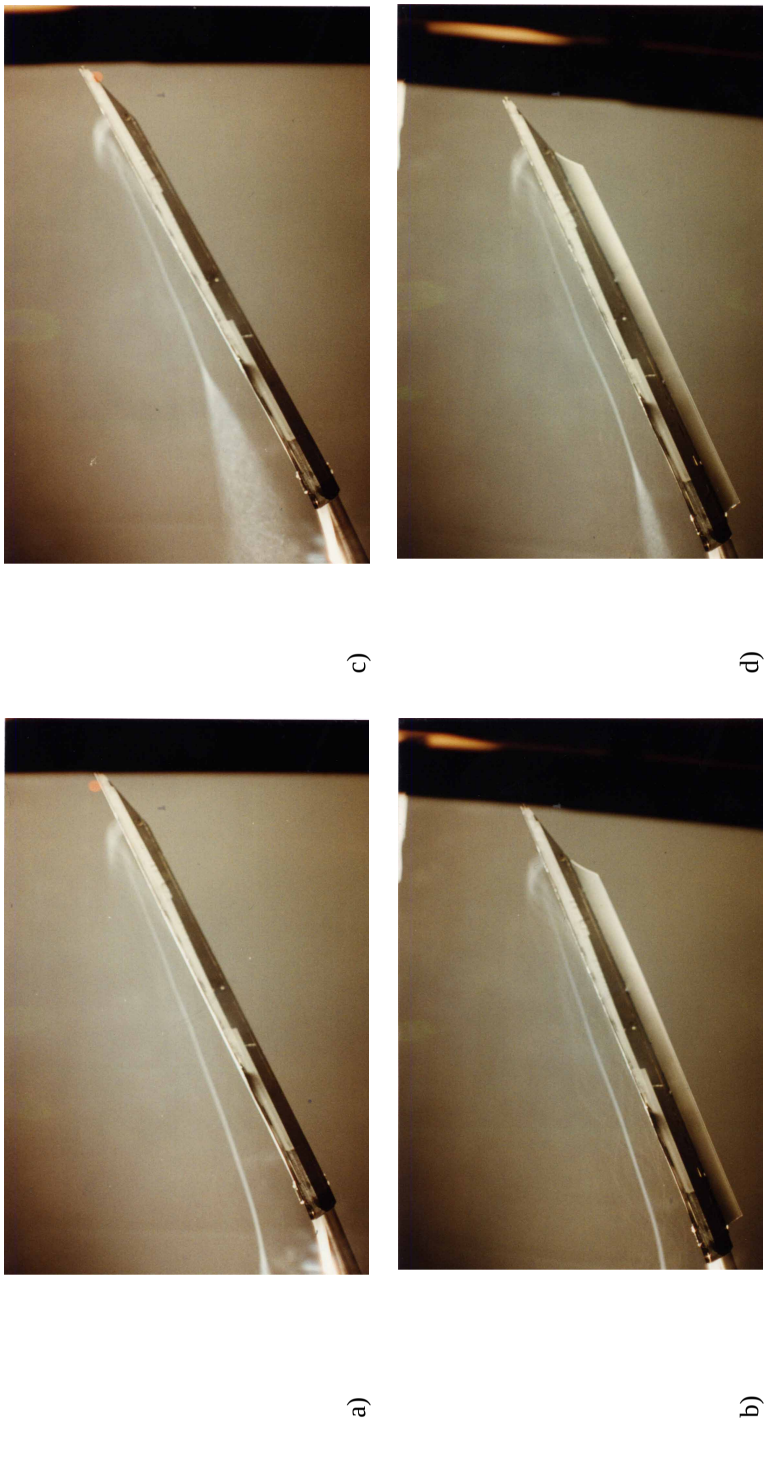


Figure 4.7: Flow Visualization of the effects of cavity flaps. a) No flaps, α=28°, b) Cavity flaps installed, α=28°, c) no flaps, α=30°, d) flaps installed, α=30°. Model: Black (1-meter chord) Tunnel: Virginia Tech Stability Wind Tunnel.

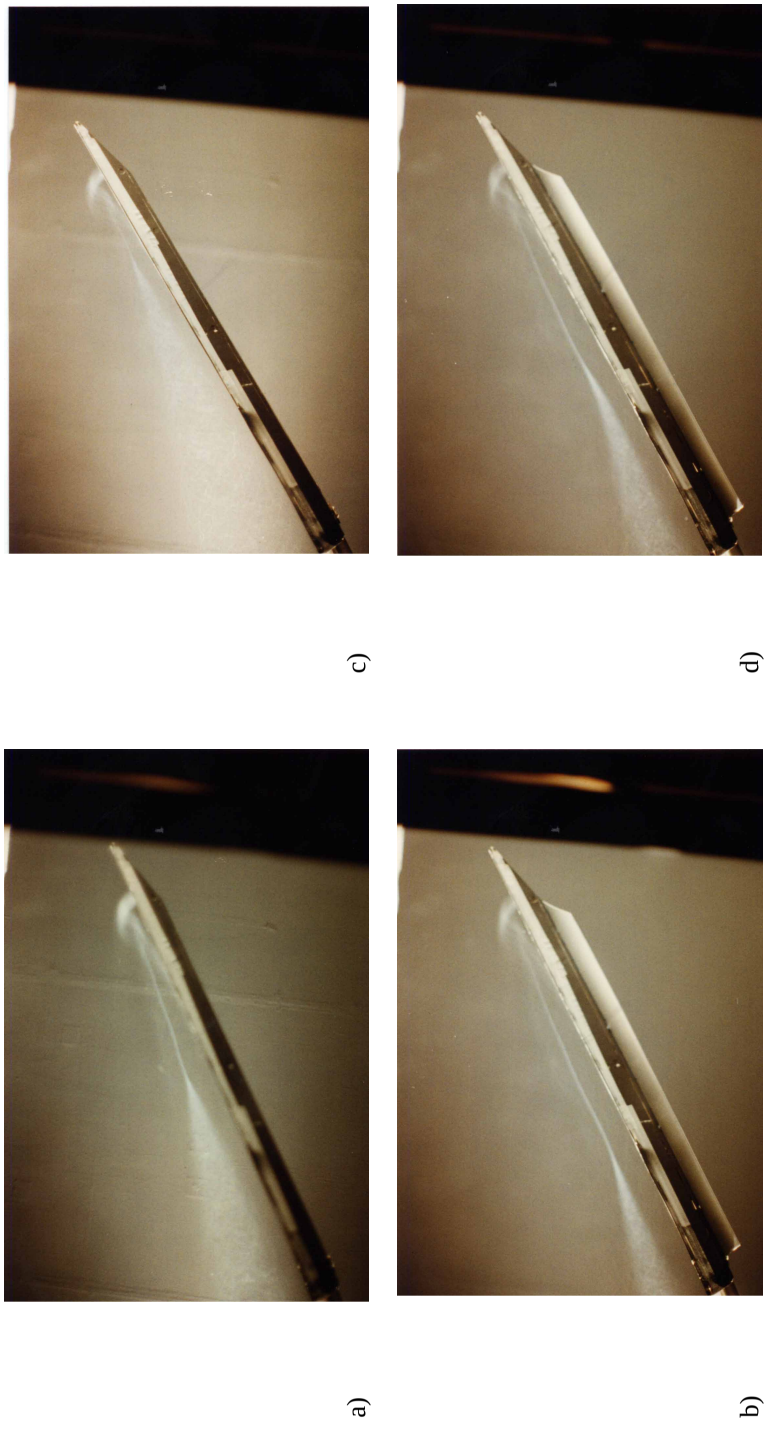


Figure 4.8: Flow Visualization of the effects of cavity flaps. a) No flaps, α=32°, b) Cavity flaps installed, α=32°, c) no flaps, α=34°, d) flaps installed, α=34°. Model: Black (1-meter chord) Tunnel: Virginia Tech Stability Wind Tunnel.

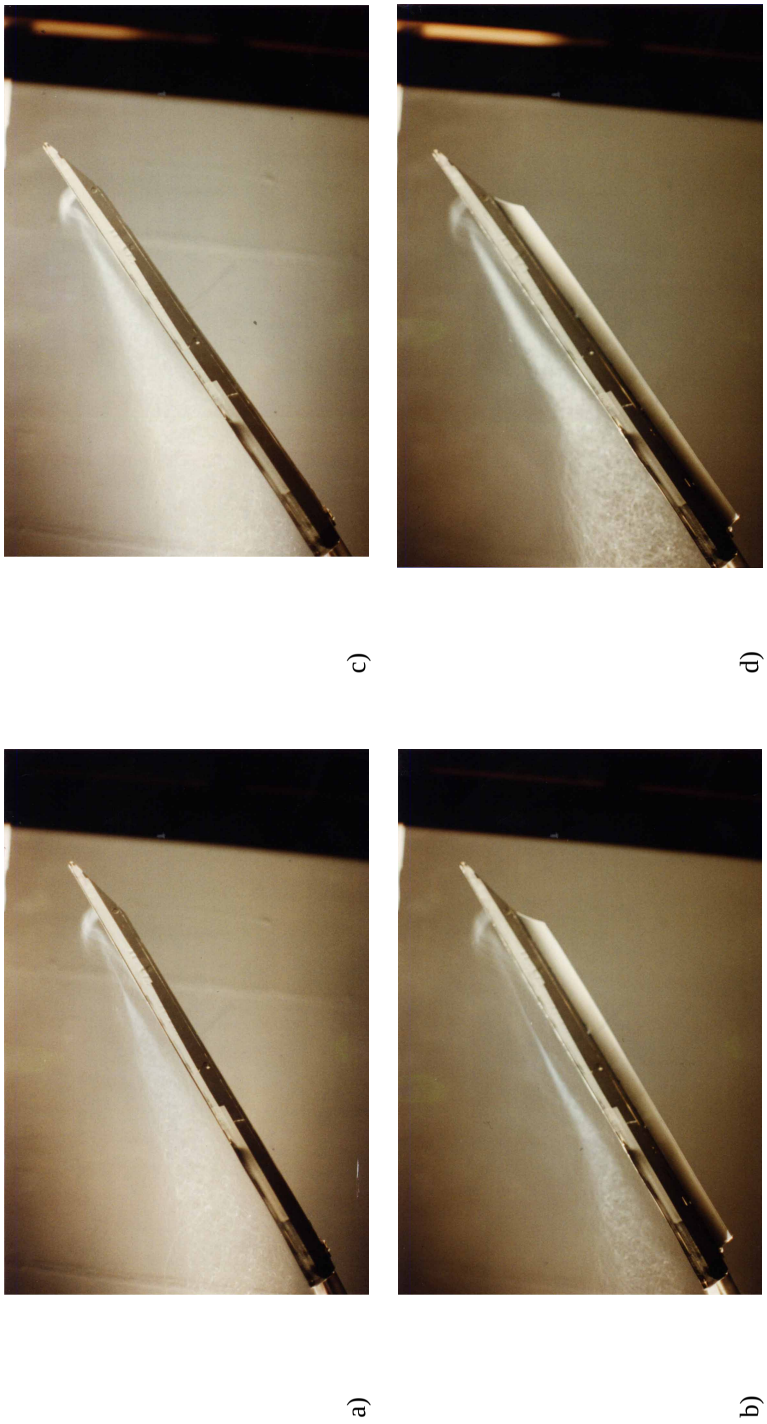


Figure 4.9: Flow Visualization of the effects of cavity flaps. a) No flaps, α=36°, b) Cavity flaps installed, α=36°, c) no flaps, α=40°, d) flaps installed, α=40°. Model: Black (1-meter chord) Tunnel: Virginia Tech Stability Wind Tunnel.

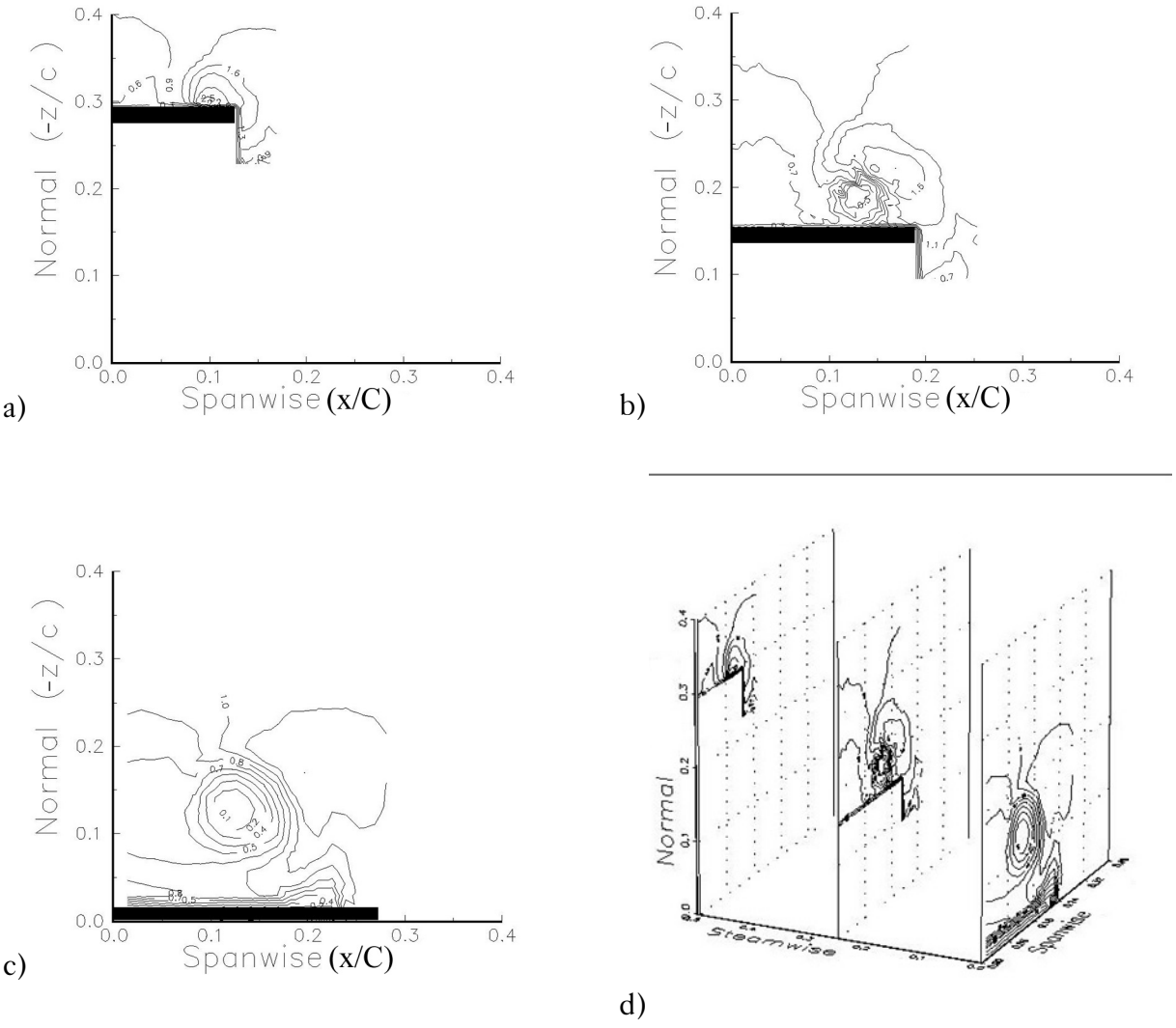


Figure 4.10: Axial velocity distribution on three planes over the plain delta wing. a) Plane A (y/C=1.0), b) Plane B (y/C=0.75), c) Plane C (y/C=0.50) and d) all three planes. The black rectangle in the first three figures represents the wing. Model: LDV (0.14-meter chord) Tunnel: ESM Water Tunnel.

pocket persists up to the trailing edge as shown in Figure 4.10a. The conical character of the apparent but in plane A, it appears that the core of the vortex is already lifting to align itself with the free-stream.

Data obtained with cavity flaps are presented in Figure 4.11. It now appears that at this angle of attack, α=35°, the flaps are capable of sustaining a coherent vortex. Axial components as high as 1.6 are observed even at the trailing edge of the wing. Moreover, it

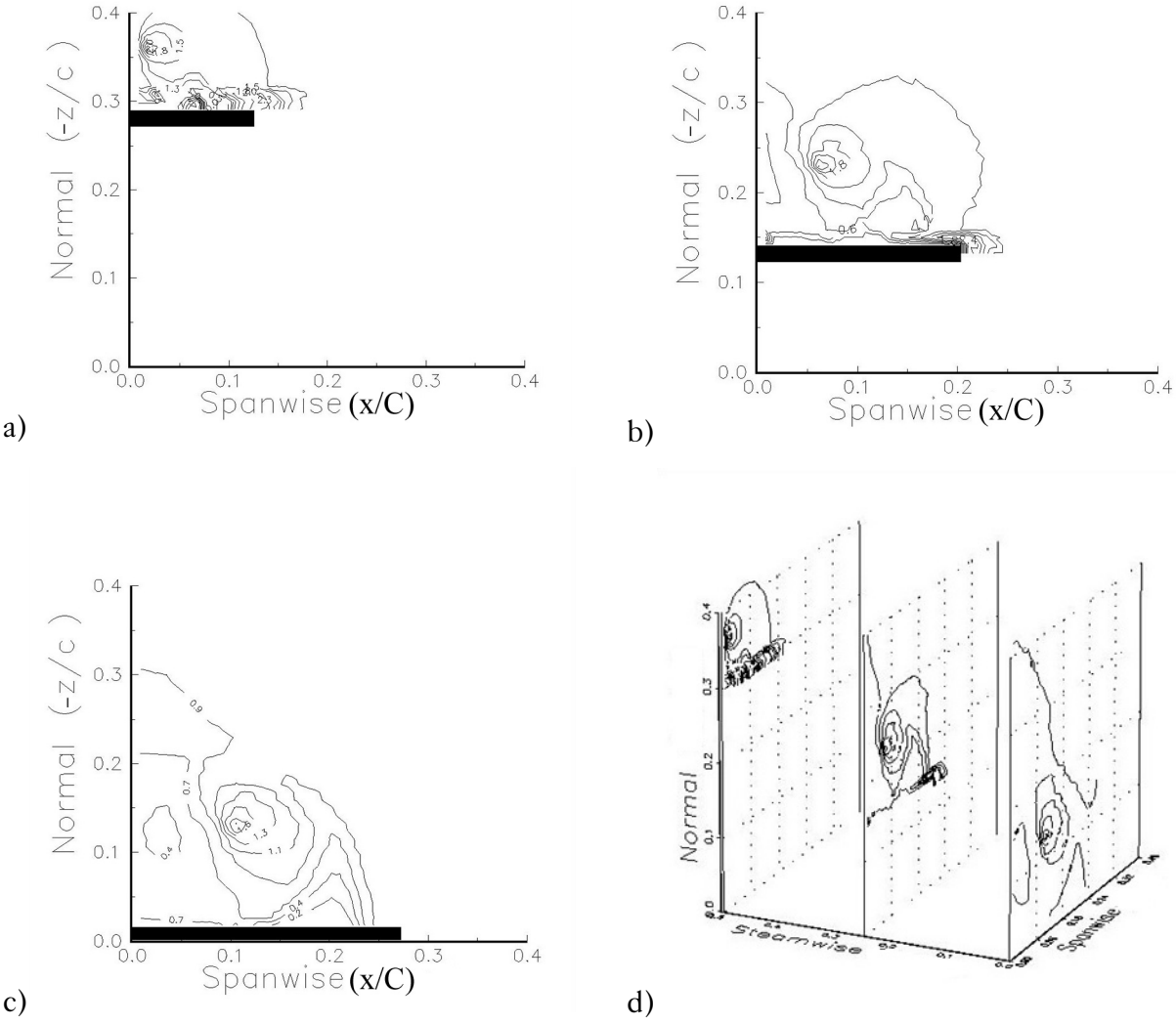


Figure 4.11: Axial velocity distribution on three planes over the cavity flap equipped delta wing. a) Plane A (y/C=1.0), b) Plane B (y/C=0.75), c) Plane C (y/C=0.50) and d) all three planes. The black rectangle in the first three figures represents the wing. Model: LDV (0.14-meter chord) Tunnel: ESM Water Tunnel.

appears that the vortex is displaced somewhat in the inboard direction.

4.1.4 Surface Pressure Distribution

The surface pressure measurements presented here were made with the Black model in the same configuration as it was for the flow visualizations. The data is intended to complement the flow visualizations and the LDV data presented earlier. The Reynolds number is 1,000,000.

The data shown in Figure 4.12 illustrate the classic behavior of the delta wing as the angle of attack increases. The curve for $\alpha=30°$ is below the curve for $\alpha=28°$, but as we approach the apex, the two curves cross at approximately y/C = 0.45. At higher angles of attack, the curves switch positions due to the presence of vortex breakdown over the planform of the wing. The resulting loss of vortex lift can be seen in the fact that subsequent angles of attack have numerically higher pressure coefficients. Figure 4.13 shows the same pressure distribution for a delta wing equipped with cavity flaps. An interesting distribution is observed. The data were obtained along the same physical line as the data of Figure 4.12, but it appears that the flaps have displaced the vortex, resulting in a trough in the pressure distribution. It can be observed, however, that as the angle of attack is increased the pressure coefficients continue to rise indicating an increase in vortex lift. It is not until an angle of attack of 36° that the remaining curves begin to drop. The peculiar trough in the domain $0.25 < y/C < 0.55$ may be due to a displacement of the core in the inboard direction and therefore away from the line of measurement.

4.2 The Performance of Cavity Flaps in Unsteady Flow

4.2.1 Continuous Deployment

Figure 4.14 shows the same distribution as Figure 4.12 but for an unsteady pitchup maneuver. As the angle of attack increases, the pressure coefficients continuously drop, illustrating the influence of the unsteady nature of the flow. It is not until an angle of attack

of 42.78°, the end of the motion schedule, that the curves begin to indicate a leveling of the pressure. There are two lines of data for that angle, one at motion's end, the other is the

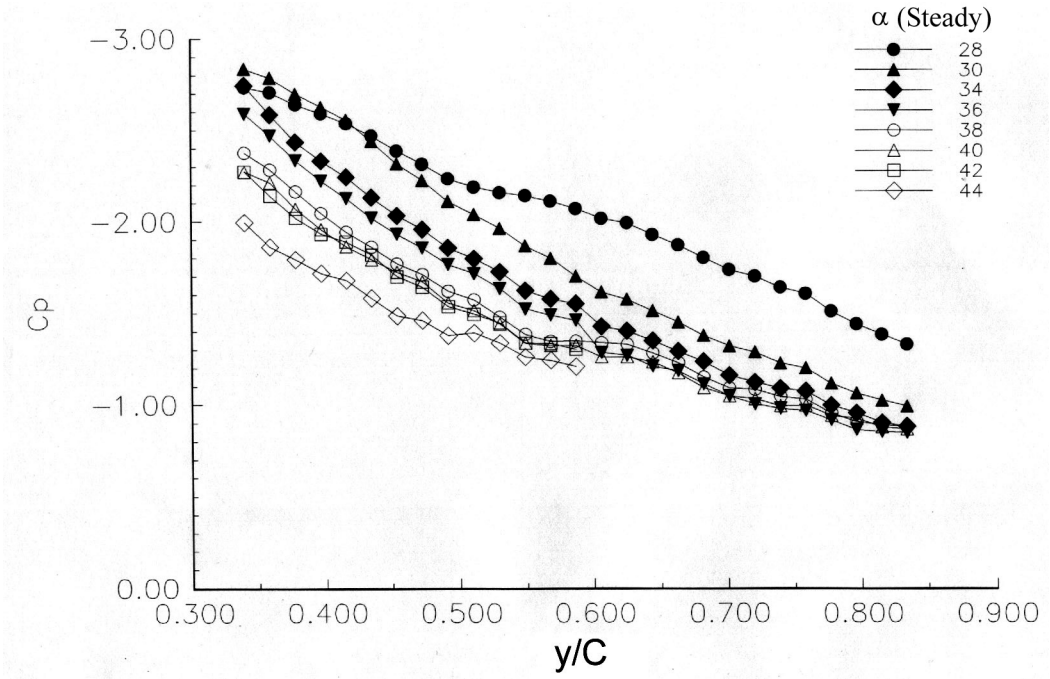


Figure 4.12: Surface Pressure Distribution along an axial line. Steady Flow. Plain Delta Wing. Re = 1,000,000 Model: Black (1.0-meter chord) Tunnel: Virginia Tech Stability Tunnel.

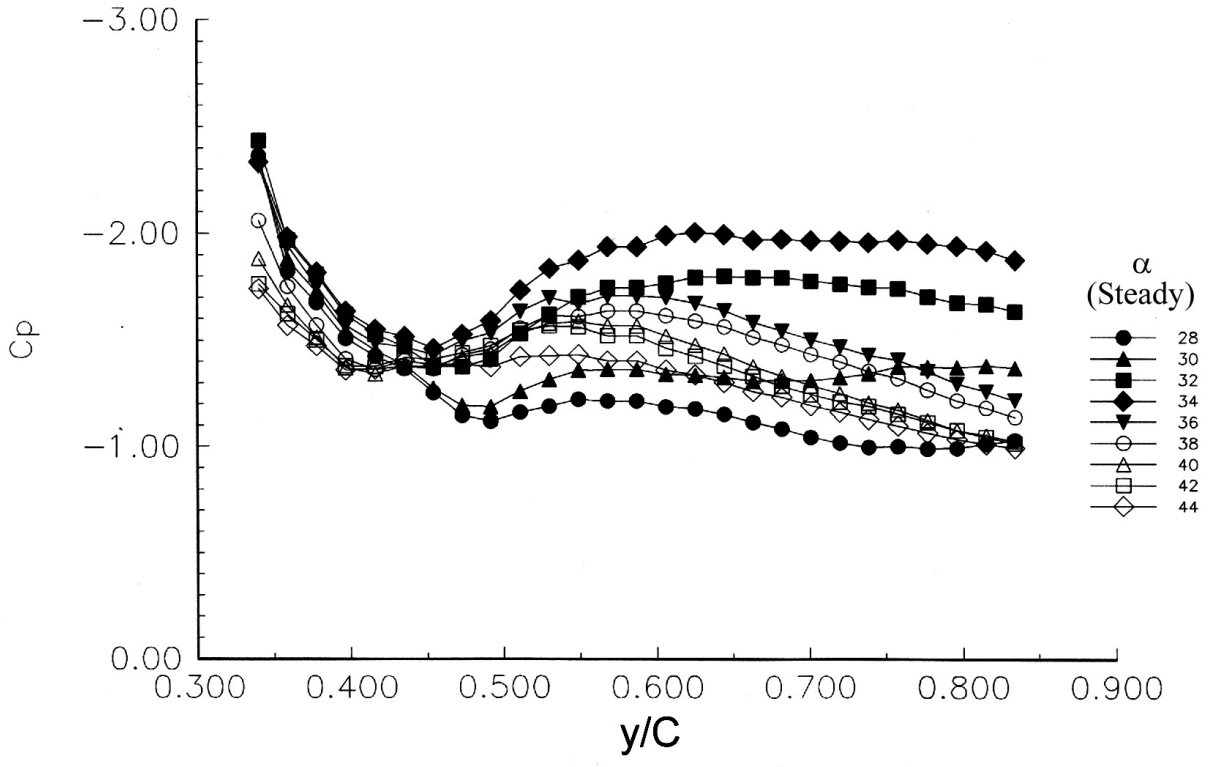


Figure 4.13: Surface Pressure Distribution along an axial line. Steady Flow. Delta Wing with Cavity Flaps. Re = 1,000,000 Model: Black (1.0-meter chord) Tunnel: Virginia Tech Stability Tunnel.

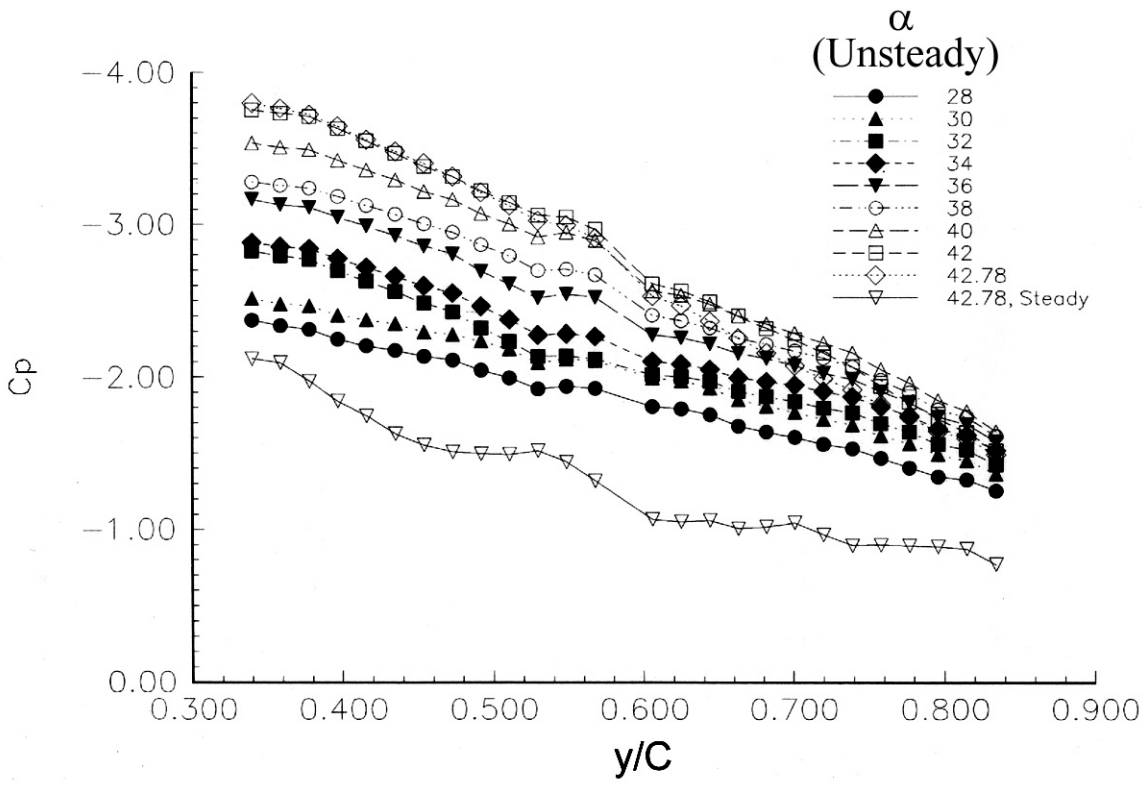


Figure 4.14: Surface Pressure Distribution along an axial line. Unsteady Flow. Plain Delta Wing. Re = 1,000,000 Model: Black (1.0-meter chord) Tunnel: Virginia Tech Stability Tunnel.

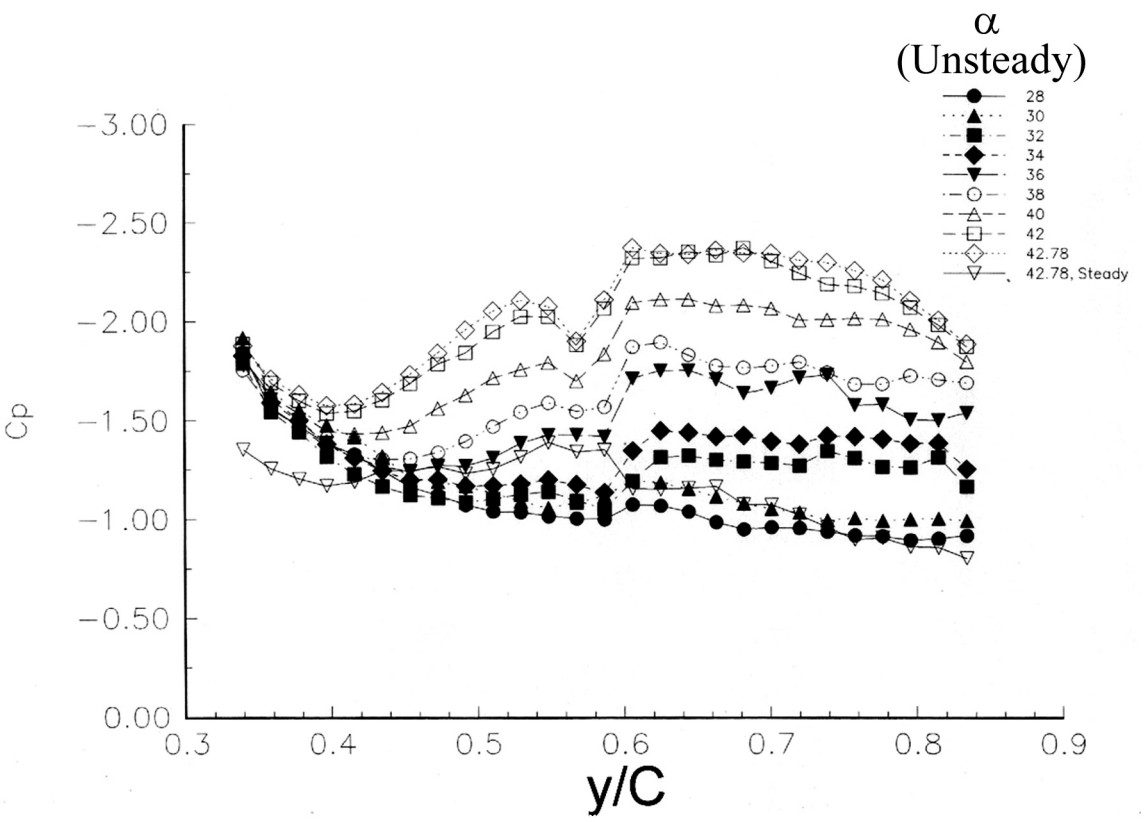


Figure 4.15 Surface Pressure Distribution along an axial line. Unsteady Flow. Delta Wing with Cavity Flaps. Re = 1,000,000 Model: Black (1.0-meter chord) Tunnel: Virginia Tech Stability Tunnel.

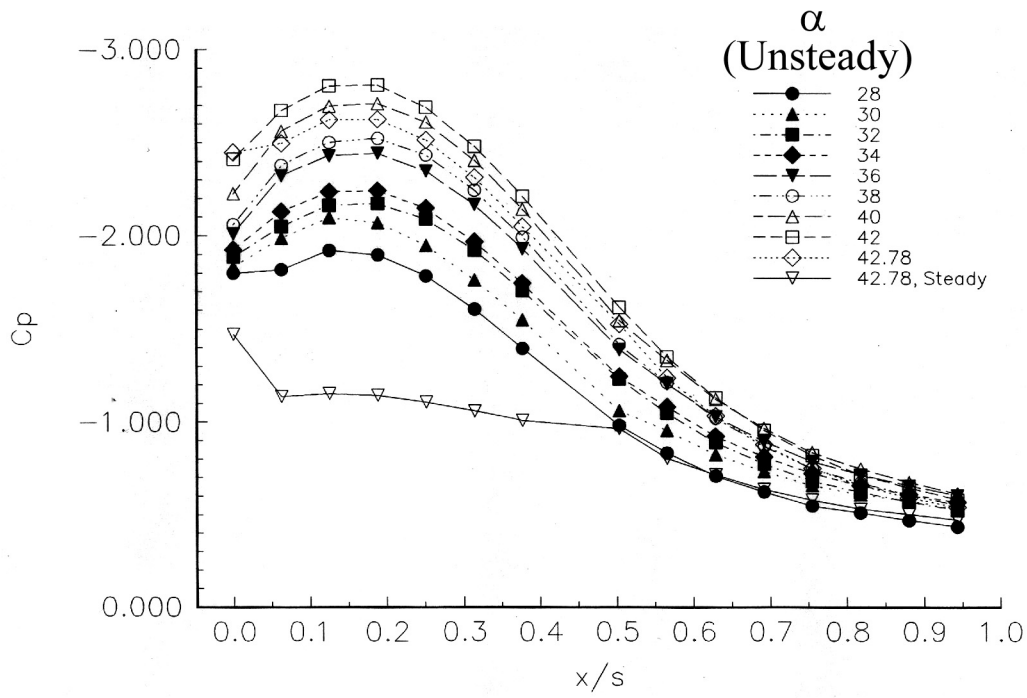


Figure 4.16: Surface Pressure Distribution in a Cross-Flow plane at y/C=0.61. Unsteady Flow. Plain Delta Wing. Re = 1,000,000 Model: Black (1.0-meter chord) Tunnel: Virginia Tech Stability Tunnel.

steady flow distribution obtained after the resting period, a time equal to two periods of the pitchup motion. Figure 4.15 is an unsteady version of Figure 4.13, obtained over the delta wing with cavity flaps. Here the curves never switch direction, each has lower pressure coefficient than the previous, all the way to the highest angle of attack.

In Figure 4.16 we display data for a pitchup motion, but for the surface pressure distribution in a cross flow plane located at x/L = 0.61, for a plain delta wing. The same trend is evident. The curves show evidence of a loss of vortex lift near the end of the motion. Figure 4.17 presents similar data for a wing with cavity flaps. Again there is no evidence of any loss of vortex lift. The curves never reverse as the angle of attack increases.

Figure 4.18 presents the same data shown in Figure 4.16 and 4.17, but in a different format. In Figures 4.16 and 4.17, data were selected from the 900 time instances sampled during the motion so that the data correspond to when the wing is at an integral, even angle of attack. Ten curves are selected to examine what happened during the time the 900 samples were taken. In Figure 4.18 every third data set during the pitchup motion is represented. A matrix of values was created, containing pressure port versus angle of attack. The pressure coefficient is assigned a color based upon its value. The resulting image shows the evolution of the surface pressure as a function of angle of attack, or as a function of time. The fact that the pressure coefficient in the plain-wing case has peaked and is beginning to decay is shown as the "island" of red in the upper left-hand corner. In the case of the wing with deployed flaps the two yellow sections show evidence of merging and a new peak just beginning to form as the motion ends.

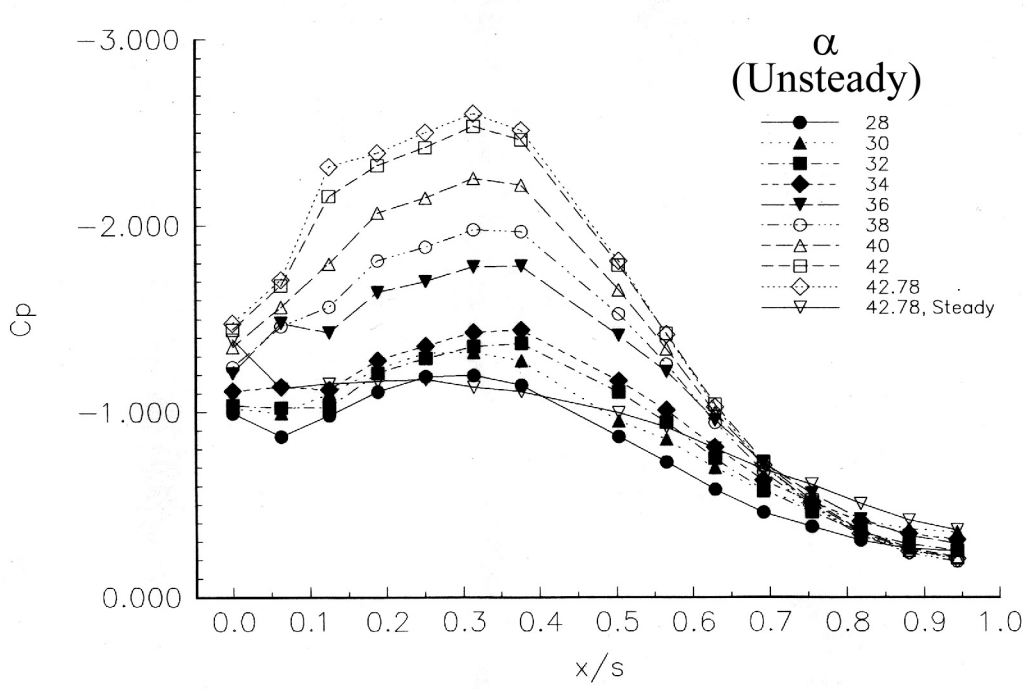


Figure 4.17: Surface Pressure Distribution in a Cross-Flow plane at y/C=0.61. Unsteady Flow. Delta Wing with Cavity Flaps. Re = 1,000,000 Model: Black (1.0-meter chord) Tunnel: Virginia Tech Stability Tunnel.

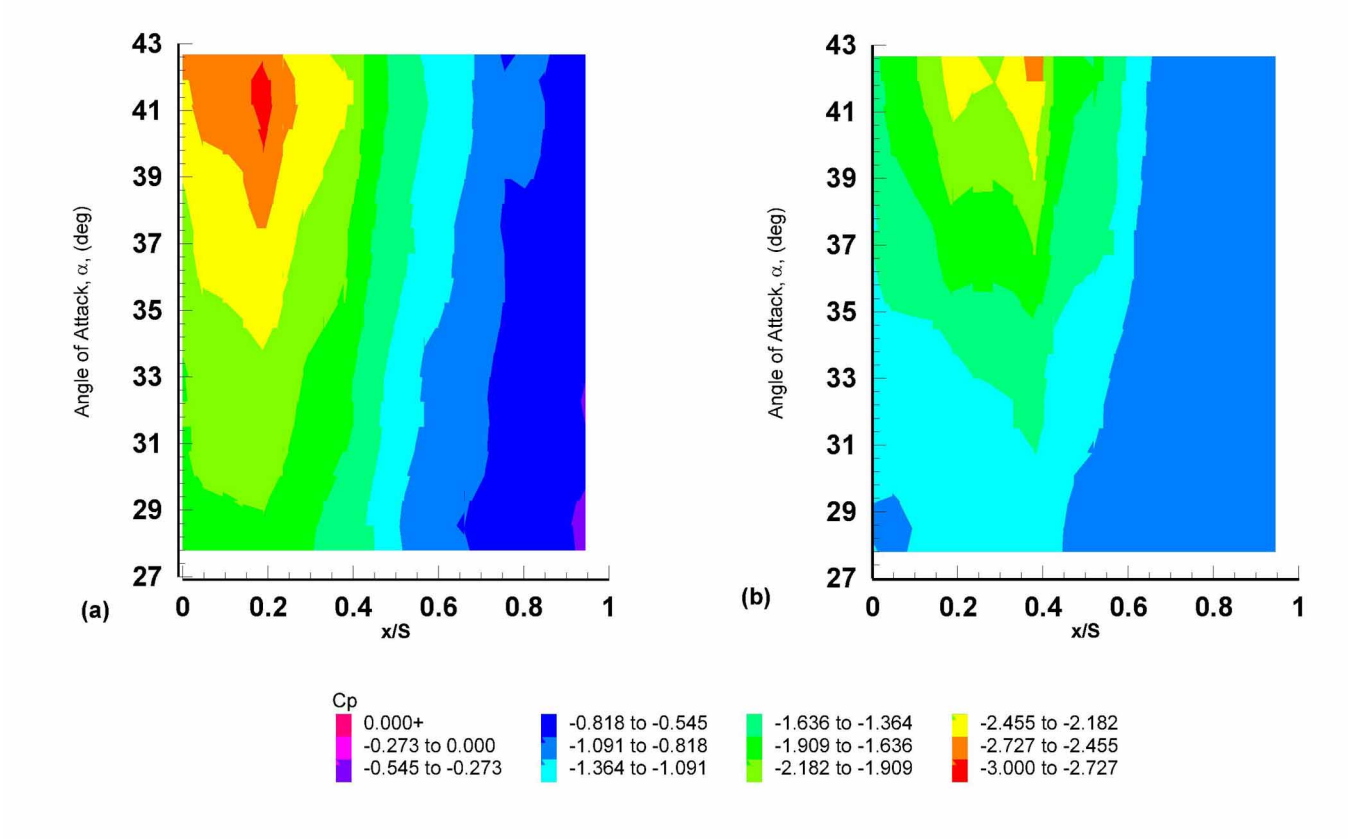


Figure 4.18: Surface pressure distribution in a cross-flow plane at y/C = 0.61 Unsteady flow, a) plain delta wing, b) delta wing with cavity flaps. Re = 1,000,000 Model: Black (1.0-meter chord) Tunnel: Virginia Tech Stability Tunnel.

4.2.2 Experimental Conditions for Cavity Flap Deployment during a Maneuver

Experiments involving cavity flap deployment were conducted in two facilities, namely the Virginia Tech Stability Wind Tunnel and the ESM Wind Tunnel. This permitted testing over a range of Reynolds numbers from 10^5 to 10^6.

In the Stability Tunnel the Black model was equipped with a set of deployable cavity flaps. Two Bimba 1.125-inch bore pneumatic actuators were installed in the model. A clevis and linkage connect the actuator to a lever arm, which is connected directly to one of the flaps. A hole was machined through the wall of the model to allow the lever arm to pass through and connect to the flaps. Mechanical drawings for the flaps and all the linkage parts are contained in Appendix A. The flaps themselves are hinged along the bottom of the model and when not deployed, are stowed flush along the side of the model. The cross section of the wing is virtually unchanged with the flaps stowed. Photographs of the flaps deployed and stowed on the Black model can be seen in Figure 4.19.

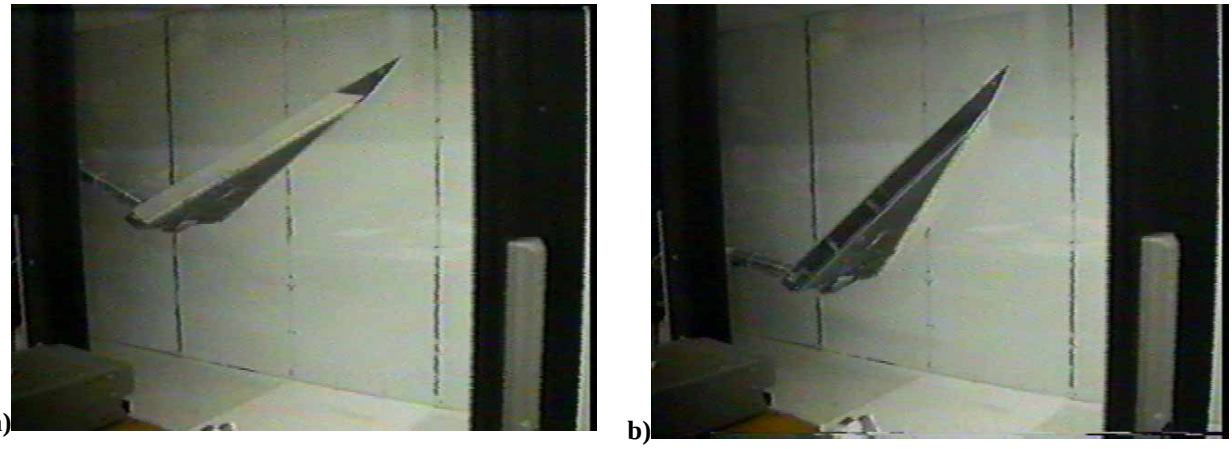


Figure 4.19: Video captured images of the Black model mounted on the DyPPiR with its cavity flaps, (a) stowed, and (b) deployed.

The pneumatic hoses that feed the actuators come out of the model through a hole in the trailing end of the model. The hoses are then secured to the sting and brought back out of the tunnel to the control valves. The control valve assembly consists of a bank of three-way

solenoid valves. Opening one valve deploys a flap, while another controls retracting of the flap. Each flap has its own complete set of valves allowing deployment independently of the other flap. The valves are interfaced to the same computer that controls the DyPPiR through a series of buffered relays, the Delta Wing Valve Control. This allows a single file to describe the motion of the DyPPiR, and hence the motion of the model as well as the deployment of the flaps. A separate computer is triggered by the DyPPiR computer to acquire the surface pressures from the ESP. The valve bank can be seen in Figure 4.20 and the Delta Wing Valve Control can be seen in Figure 4.21

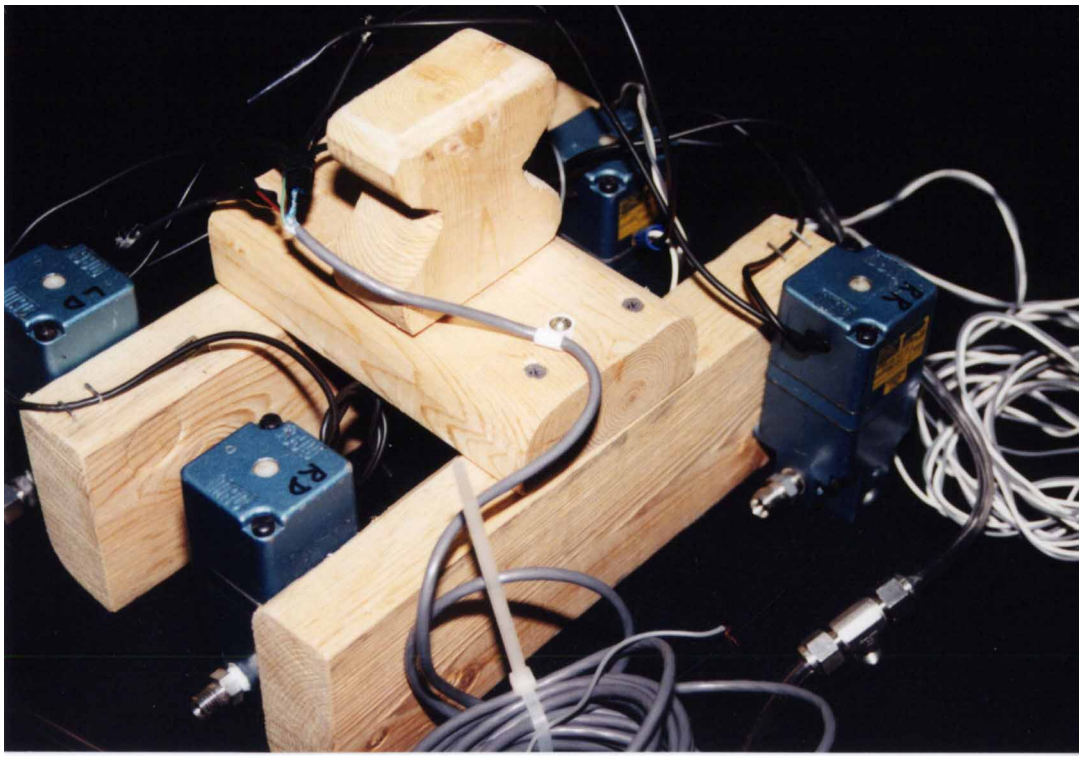


Figure 4.20: The Valve Bank used by the pneumatic system to deploy the flaps.

Figure 4.21: The Delta Wing Valve Control

In order to determine the deployment time of the flaps, the flaps themselves are instrumented. Mounted on each actuator clevis is an optical sensor. Underneath the sensor, on the floor of the model, a uniform bar code is placed. As the actuator moves to deploy the flap, the sensor passing over the bar code generates an oscillatory signal, the highest voltage corresponding to the middle of a black strip, the lowest corresponding to the middle of the white strip. This signal is converted to a TTL-compatible digital signal by a circuit mounted inside the model, and leaves the model along the same path as the pneumatic hoses. By sampling the output of the flap encoder circuit during the motion, the time of flap deployment was determined.

Dynamic motions in the ESM Wind Tunnel are accommodated by using a pitch-up mechanism, which consists of U-shaped swing mounted on supports independent of the tunnel. A schematic of this system is shown in Figure 4.22. The sting is attached to the swing and enters the tunnel through a slot in the tunnel floor. The slot is filled with brushes to allow the sting to more freely fore and aft, but yet keep ambient air from entering the tunnel. A Linear-Variable Differential Transformer (LVDT) is attached to the swing to provide

accurate angle of attack information. The swing motion is controlled by a variable speed DC motor.

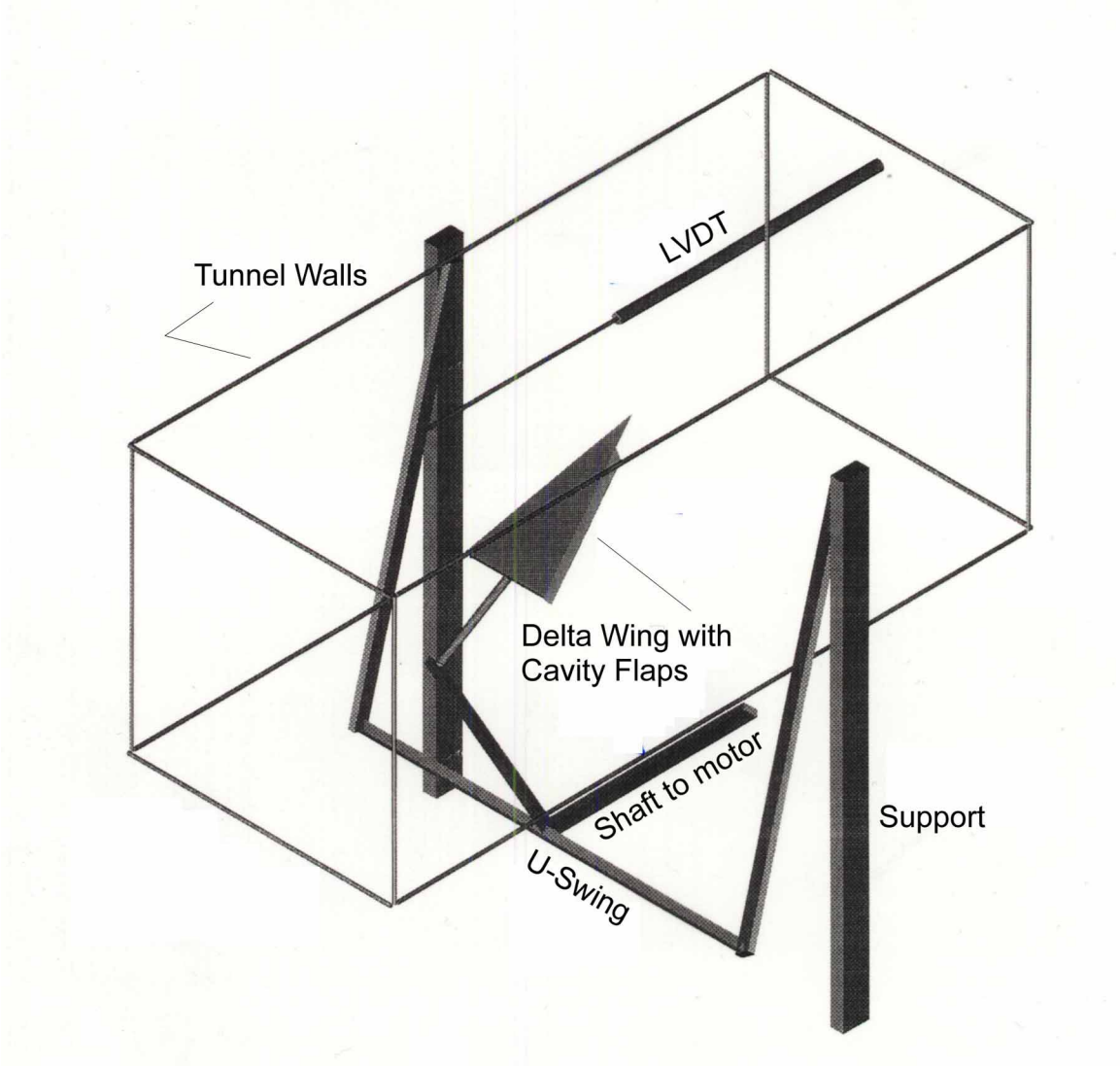


Figure 4.22: Schematic of the pitch-up mechanism for the ESM Wind Tunnel.

Two computers are utilized to control the dynamic motions. An ISA PC acts as a master with a PS/2 Model 60 acting as its slave. The model 60 is responsible for regulating the speed of the DC motor. One D/A channel of a DT 2905 Data Acquisition Card is used for

this. This voltage passes through an optical isolator and then to the motor controller. The other D/A line is used to set the level for a comparator circuit. It was buffered by a 1:1 Burr Brown Instrurnentation Amplifier. This voltage is compared with the voltage from the LVDT and provides a trigger for the flap deployment. The ISA machine is responsible for acquiring the data from the ESP, via an ESP/PC interface card by Aeroprobe Corporation, and from the LVDT, via an RC Electronics ISC-16 Data Acquisition Card. The ISA machine also controls the start position of the model and the start of the pitch-ups by communicating with the Model 60 via their parallel ports. Data acquisition and pitch-up start are governed by the same trigger. All data is stored on the ISA machine. A block diagram of the interaction of the two computers and the rest of the instrumentation is shown in Figure 4.23.

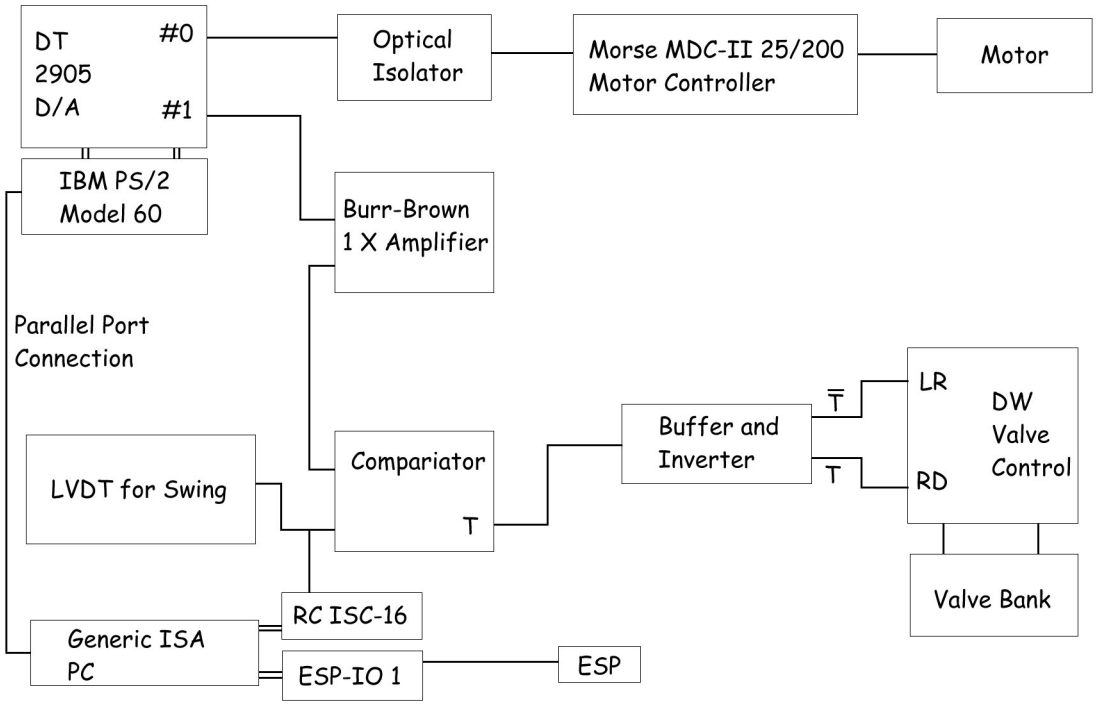


Figure 4.23: Block Diagram of the computer/instrumentation setup in the ESM Wind Tunnel. Double lines indicate connections internal to the PC.

The model utilized in the ESM Wind tunnel is the Red model, with a 0.284-meter chord. The model is mounted on the end of Bimba 9/16" bore double acting pneumatic cylinder, which doubles as the sting for the model. Both flaps are controlled simultaneously. Extending the cylinder rod causes the flaps to retract, via control cables. Withdrawing the cylinder allows torsion springs to deploy the flaps. The same control value system used in the Stability Tunnel is also utilized here. Photographs showing the Red model, the actuator/sting and the control cables are presented in Figure 4.24.

In both facilities, the ±20-inches of water ESP-32 was used to measure the surface pressures. To minimize tube length in the Stability Tunnel, the ESP is mounted inside the model. In the ESM Wind Tunnel, it is mounted on the sting.

4.2.2 Surface Pressure Distributions for Flap Deployment during a Maneuver

The surface pressure distribution over the Black model for a pitch-up motion from 28° to 53° in 1.0 seconds was recorded at a rate of 900 samples/second. The reduced frequency of this motion is 0.0089 and Reynolds number of 1,300,000. Two cases were run for these conditions, cavity flap deployment at 32° and cavity flap deployment at 36°. From the surface pressure distribution in the crossflow plane at y/C=0.61 for the 32° flap deployment presented in Figure 4.25, it appears that at first the effect of the flap deployment is actually detrimental and reduces the level of suction at α=42°, compared to the baseline case (Figure 4.16). However when the wing reaches an angle of attack of 46°, the situation is improved of the baseline case. We see first a delay in the effect of deploying the flap, then a negative effect, followed by a positive effect. This is attributed to the fact that as the flap deploys, some of the vorticity that is being shed is entrained in the creation of the cavity vortex, temporarily starving the main vortex system of fed vorticity. Once the cavity vortex is fully developed, more of the vorticity being shed goes to the main system duplicating the effect that cavity flaps have for steady angles of attack. Deploying the flaps when the wing is passing the angle of attack of 36°, data presented in Figure 4.26, improves the situation

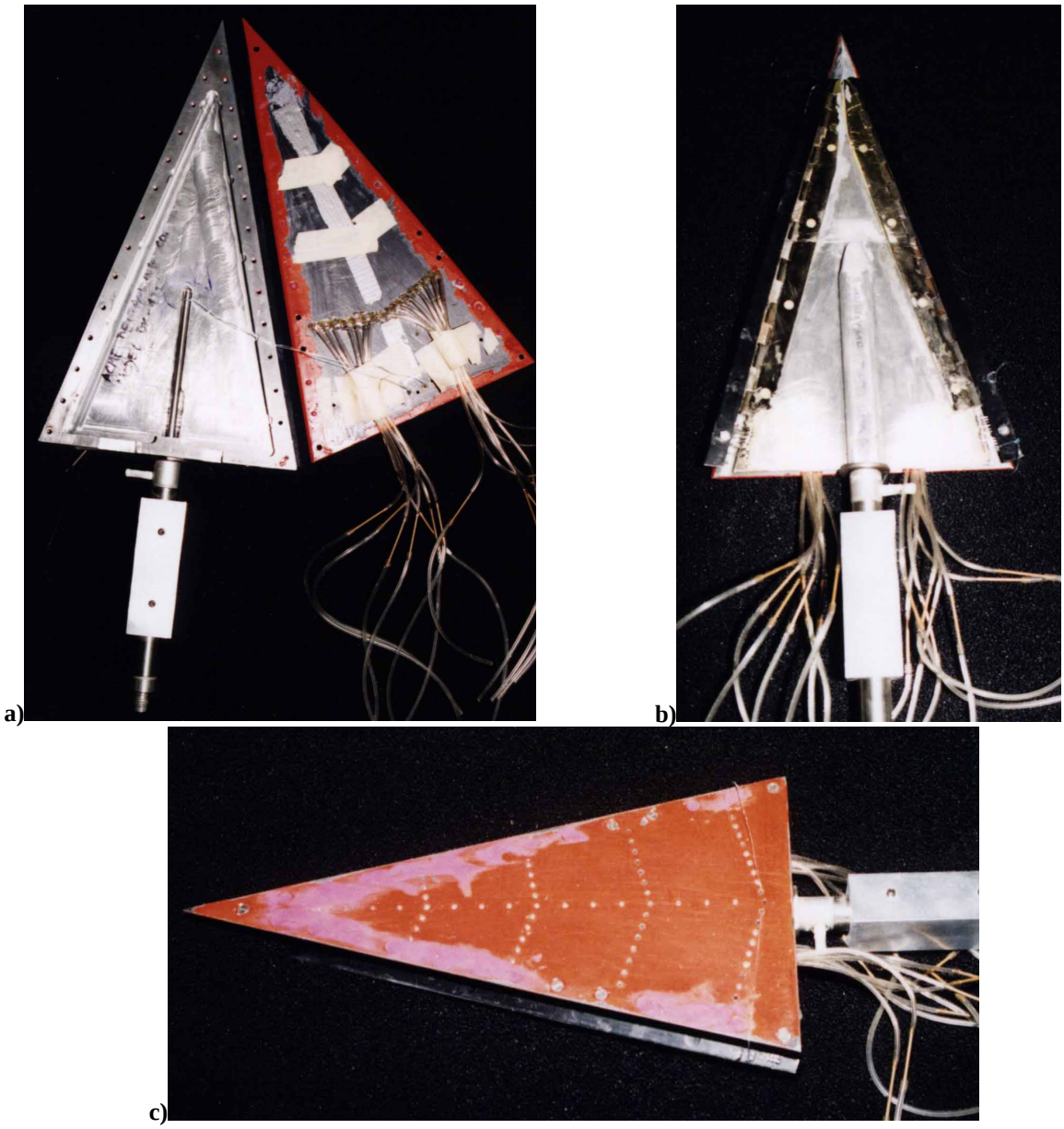


Figure 4.24: Various Views of the Red model, (a) inside the model showing actuator rod and tubing attachments, (b) bottom of the model, (c) top of the model. The sting is actually the Bimba pneumatic actuator and the block attached to provided and area to mount the ESP-32. V-shaped crossflow port lines are located with their origins at y/C=0.42, 0.60, 0.78, 0.96

significantly for higher angles of attack, α=46°, if compared to the 32° flap deployment, but the overall effect does not show much departure form the baseline configuration.

Similar results can be seen in the ESM Wind Tunnel. In this facility the Reynolds number is 187,000 and the reduced frequency is the same as in the Stability Tunnel tests, k=0.0089, y/C = 0.60. Figure 4.27 is the baseline case with no flaps deployed. Figure 4.28 is the case of 32° flap deployment. Again, we see an initial detrimental effect on the surface pressure distribution followed by a brief increase in the amount of suction generated.

Presented in Figure 4.28 is a survey of the effect of different reduced frequencies at constant Reynolds number, in this case 1,300,000 and y/C = 0.61. Six different reduced frequencies were tested namely, 0.0089, 0.0125, 0.0161, 0.0197, 0.0233, and 0.0267. It is evident from examining the data presented that the higher the reduced frequency, the more likely it is that the suction will be sustained, although some reduction in the strength of the suction peaks is noticeable. In fact the distribution of the pressure indicates that the vortex is not broken down but rather that at high angles of attack it loses some of its strength and therefore produces somewhat reduced suction.

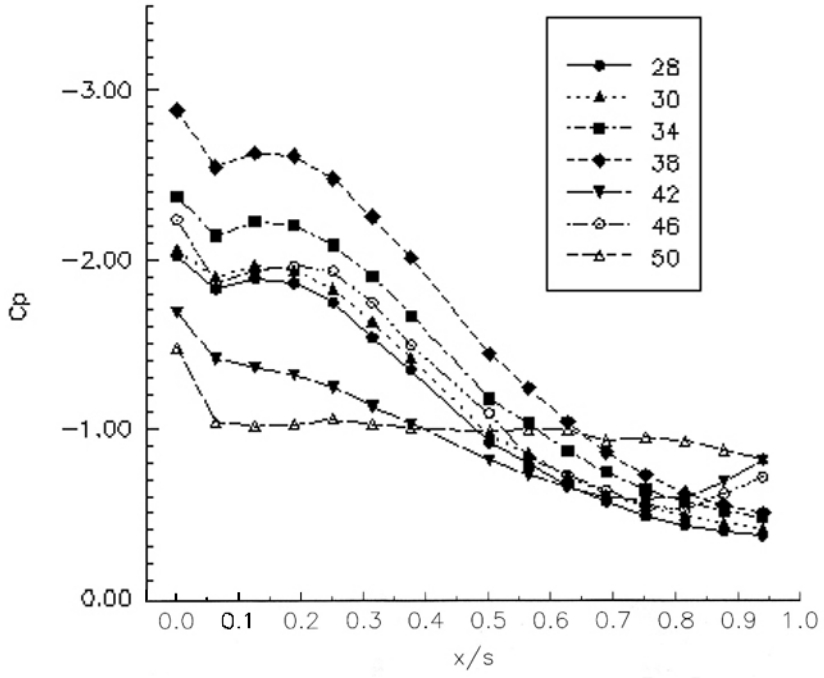


Figure 4.25: Surface pressure distribution for the Black model with flap deployment at 32°. Re = 1,300,000 k=0.0089, y/C=0.61Tunnel: Virginia Tech Stability Tunnel.

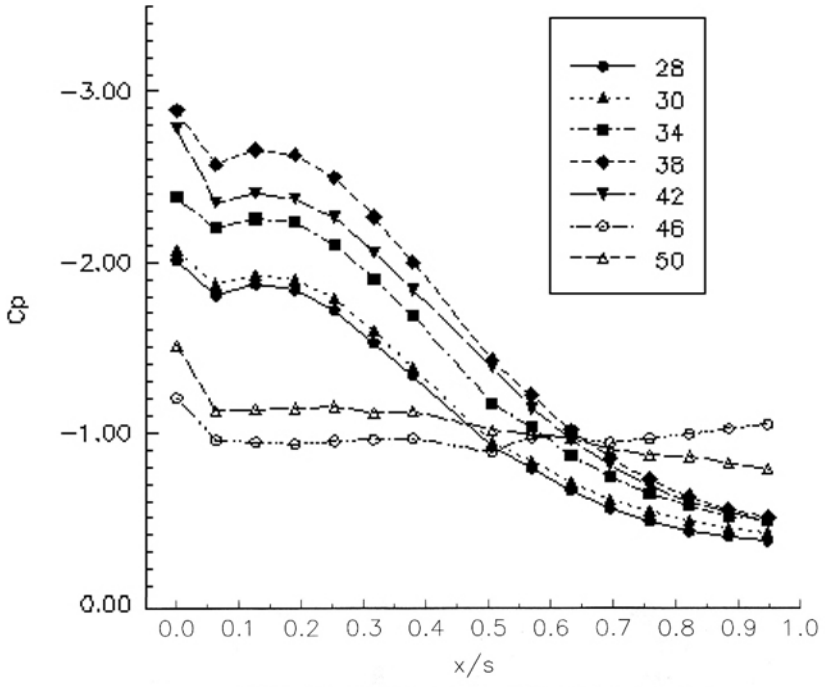


Figure 4.26: Surface pressure distribution for the Black model with flap deployment at 36°. Re = 1,300,000 k=0.0089, y/C = 0.61 Tunnel: Virginia Tech Stability Tunnel.

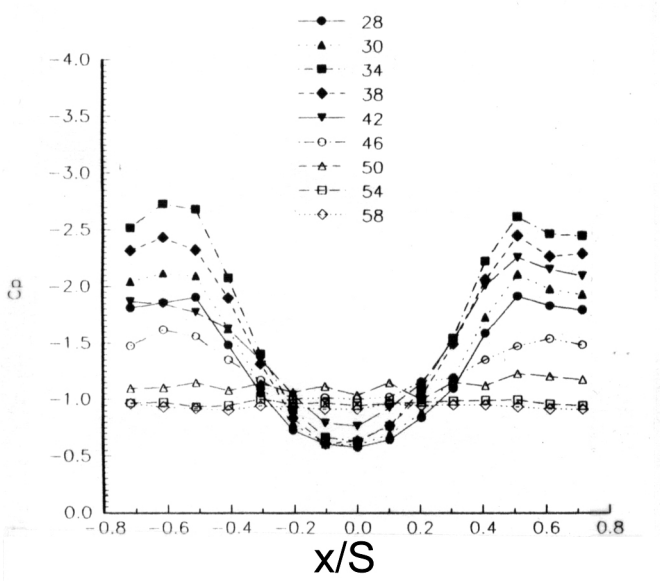


Figure 4.27: Baseline Case for the Red (0.28-meter chord) model in the ESM Wind Tunnel. Re=187,000, k = 0.0089, y/C = 0.60

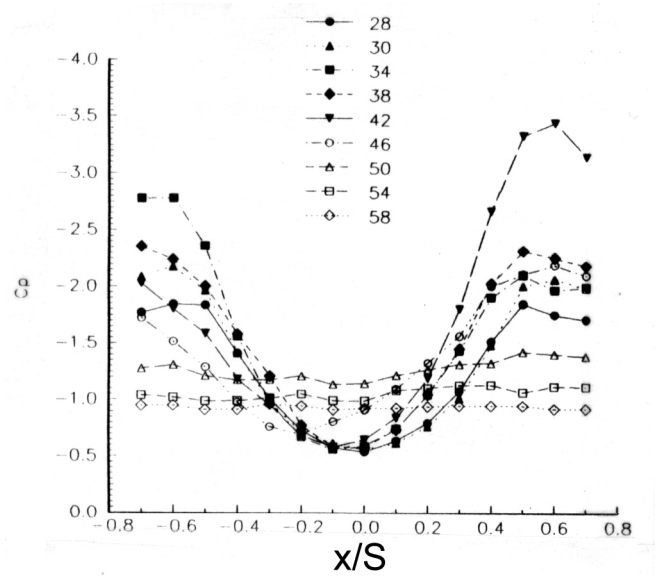


Figure 4.28: 32° Flap Deployment Case for the Red (0.24-meter chord) model in the ESM Wind Tunnel. Re=187,000, k = 0.0089, y/C = 0.60

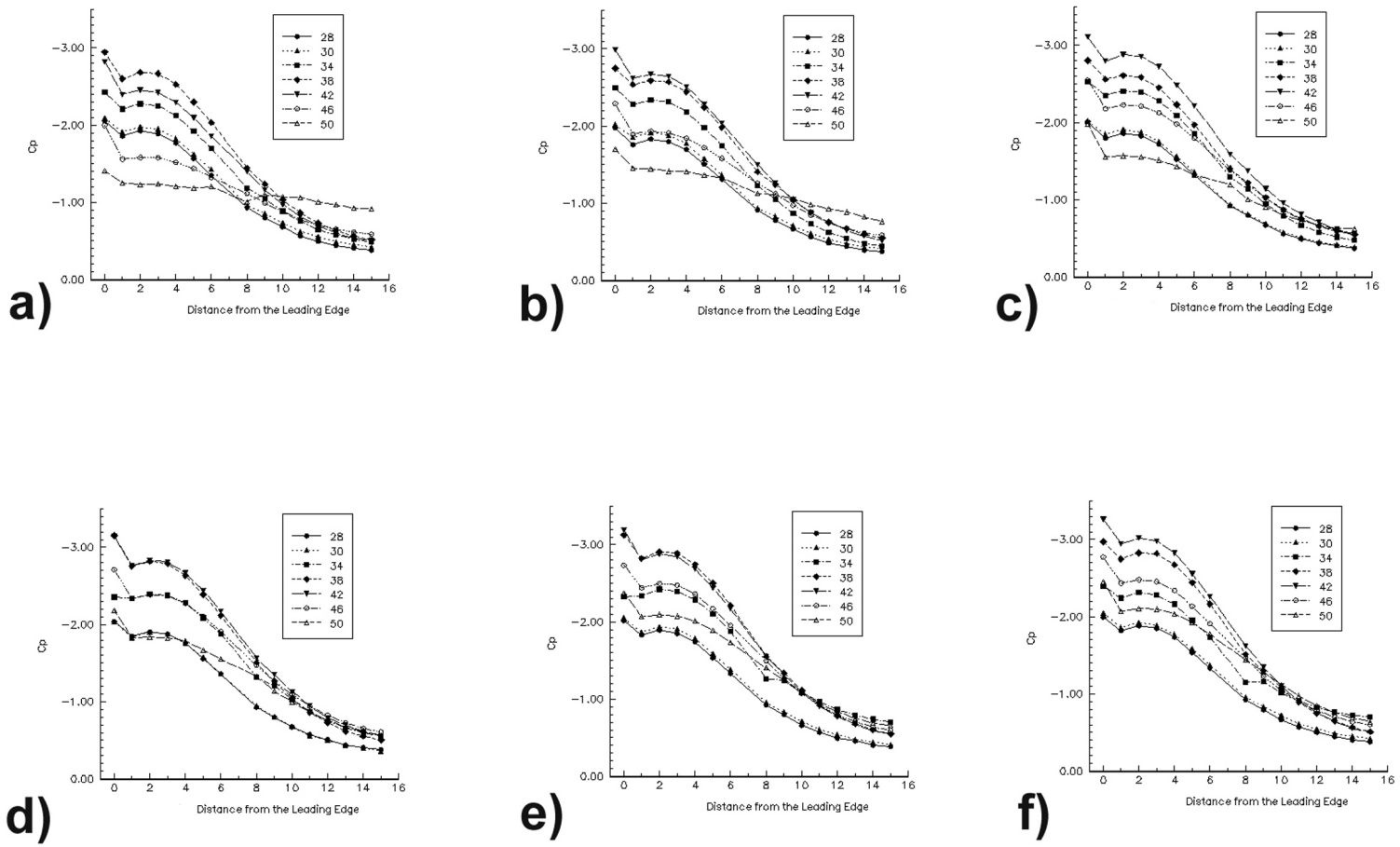


Figure 4.29: Variation of the coefficient of pressure for different reduced frequencies at constant Reynolds number, Re=1,200,00. y/C = 0.61 (a) k = 0.0089, (b) k=0.0125, (c) k=0.0161, (d)=0.0197, (e)=0.0233, and (f)=0.0267. Model: Black (1-meter chord) Tunnel: Virginia Tech Stability Wind Tunnel

CHAPTER 5: CONTROL VIA AN APEX FLAP

In this research effort, we explore the apex flap as a delta wing control surface system. A number of flap systems have been adapted to delta wings. Rao (1979), Marchman (1981), and Reddy (1980) investigated the effect of leading edge vortex flaps and demonstrated their usefulness in controlling the amount of drag a wing produces. Gursul et. al. (1995) investigated the use of leading edge flaps and variable sweep wings as a means of controlling vortex breakdown. Lamar (1986) has experimented with a single trailing edge aileron, tip mounted on a cropped delta wing. This was successful in generating a constant roll moment. Several reviews of the subject of controlling delta wings through the use of control surfaces have appeared, including Lee and Ho (1980) and Rao and Campbell (1987). The goal of a delta wing control surface system is two-fold. First, to postpone the start of the post-stall regime by delaying vortex breakdown to as high an angle of attack as possible and second once in the post-stall regime offer the ability to give some beneficial order to the wake-like broken-down vortex systems.

The need for experiments to be run at high Reynolds number in order to correctly model the physics involved in high-angle-of-attack aerodynamics is beginning to receive more attention. Ericsson (1994) has investigated a "cobra" type maneuver, i.e. a pitch-up followed by a coning motion, with reference to the vortex asymmetry on a maneuvering forebody. Ericsson finds that there is a lack of fidelity between experiments run at low Reynolds number and ones run at higher Reynolds numbers, indicating the need for experiments conducted at high Reynolds numbers.

5.1 Apex Flap Deployment during a 28° to 50° Pitchup

5.1.1 Experimental Conditions

For all the data presented in this chapter, the White model (0.67-meter chord) was utilized. The model was equipped with an apex flap, the section of the wing from the apex to a

position at 40% of the chord length was hinged to deflect down relative to the rest of the wing. This flap-angle, or the angle between the deflected flap and the wing, was held at 15° when the flap was deployed. The model was equipped with two lines of pressure taps positioned at y/C of 0.70 and y/C of 0.80. Each line is V-shaped with each side perpendicular to a leading edge of the model and has 7 pressure ports on each side of the centerline and one on the centerline. The model is shown schematically in Figure 5.1. The pressure tap spacing on the forward line (y/C = 0.70) is h/S = 0.095, where S is the length of the local semi-span and h is the spacing between the ports. The pressure tap spacing on the aft line (y/S = 0.80) is h/S = 0.102.

During an experimental run, each of the ports on one of the lines is connected by a short run of tubing to the ±10-inches of water ESP-32, which was mounted underneath the wing. During the motion of the wing, the set of 16 channels is sampled 200 times via a ESPIO-1 interface by Aeroprobe Corporation utilizing 12 bit resolution on the A/D conversion.

Deployment of the apex flap is accomplished by using a Bimba 1.125 inch bore pneumatic actuator installed underneath the model. The travel of the actuator was limited by a locking collar, so that the flap-angle could be controlled. In the present research effort, the flap-angle was 15° for all cases. Pneumatic lines are fed along the underside of the wing, secured to the sting and then routed out of the tunnel to the control valve assembly. This assembly consists of a bank of three way solenoid valves and a buffered interface to the DyPPiR Control Computer and is the same as utilized in the cavity flap experiments. During the dynamic motion of the model, one of the outputs of the control computer triggers the

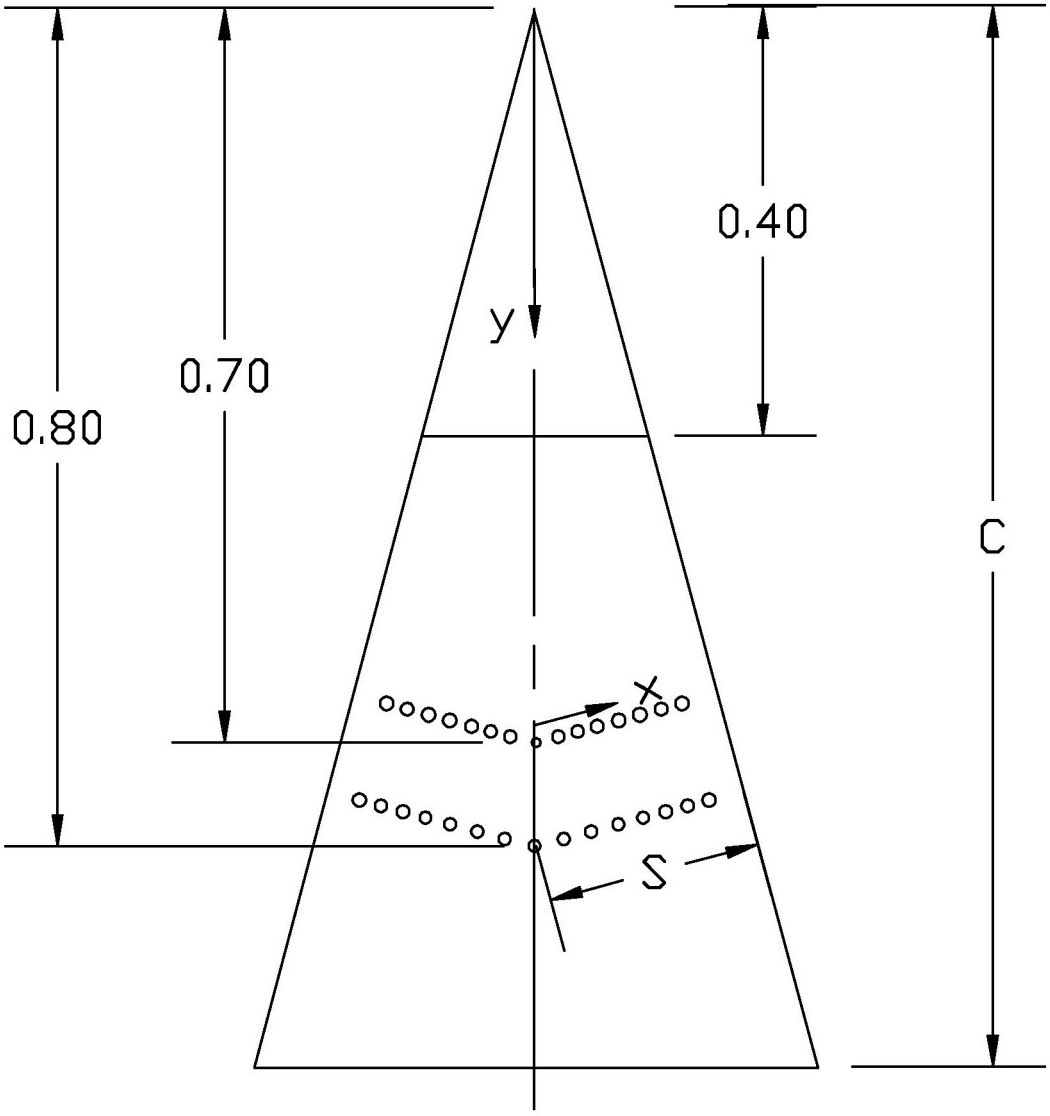


Figure 5.1: Schematic of the model used in the present research effort.

deployment of the flap. It has been estimated that there is a 0.08 second delay between the time of the trigger and the full deployment of the flap. This corresponds to a 1.75° increase in the angle of attack. To be consistent, deployment angles will be stated in terms of the

triggered time. Therefore, in a case described as a 30° flap deployment, the flap was actually fully deployed by the time the wing reaches 31.75° angle of attack.

The model is sting mounted. The sting is mated with the roll actuator of the DyPPiR so that the axis of the sting and the axis of the actuator are co-linear. There is then a 50° offset mounting bracket which is secured on the bottom surface of the wing to minimize interference effects. The ESP-32 is secured to the bottom of the wing to minimize the length of the tubing runs required. This setup can be seen in Figure 5.2. The net effect of the mounting geometry is that when the pitch actuator, and therefore the sting, is at 0° angle of attack, the model is at an angle of attack of 50°.

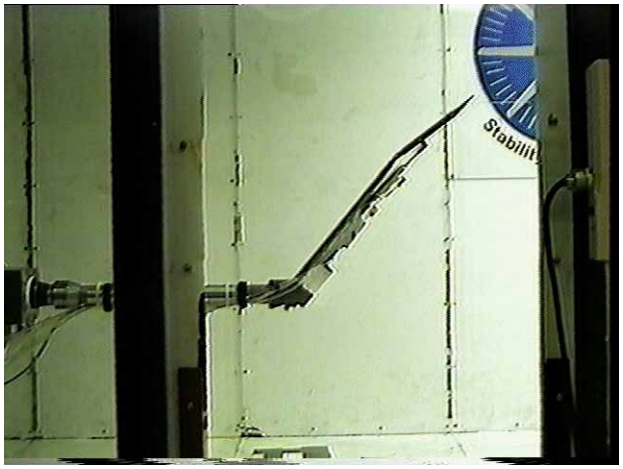


Figure 5.2: Photo of the experimental rig on the DyPPiR, showing the 50° offset mounting geometry.

5.1.2 Description of Motions

The pitch-up schedule used in this research effort is shown in Figure 5.3. It consists of a ramp function from 28° to 50° angle of attack in one second. All of the experiments that are presented here were run at a Reynolds number of 1.0×10^6, based on the chord length of the model. Hence, the reduced frequency of the pitch-up maneuver is $k = 0.0108$.

The model stays at the maximum angle of attack for 5 seconds after the motion is complete and twenty seconds separate the end of the re-initialization to the start angle of attack and the beginning of the next ensemble. For each experimental run, 30 realizations of the motion were performed.

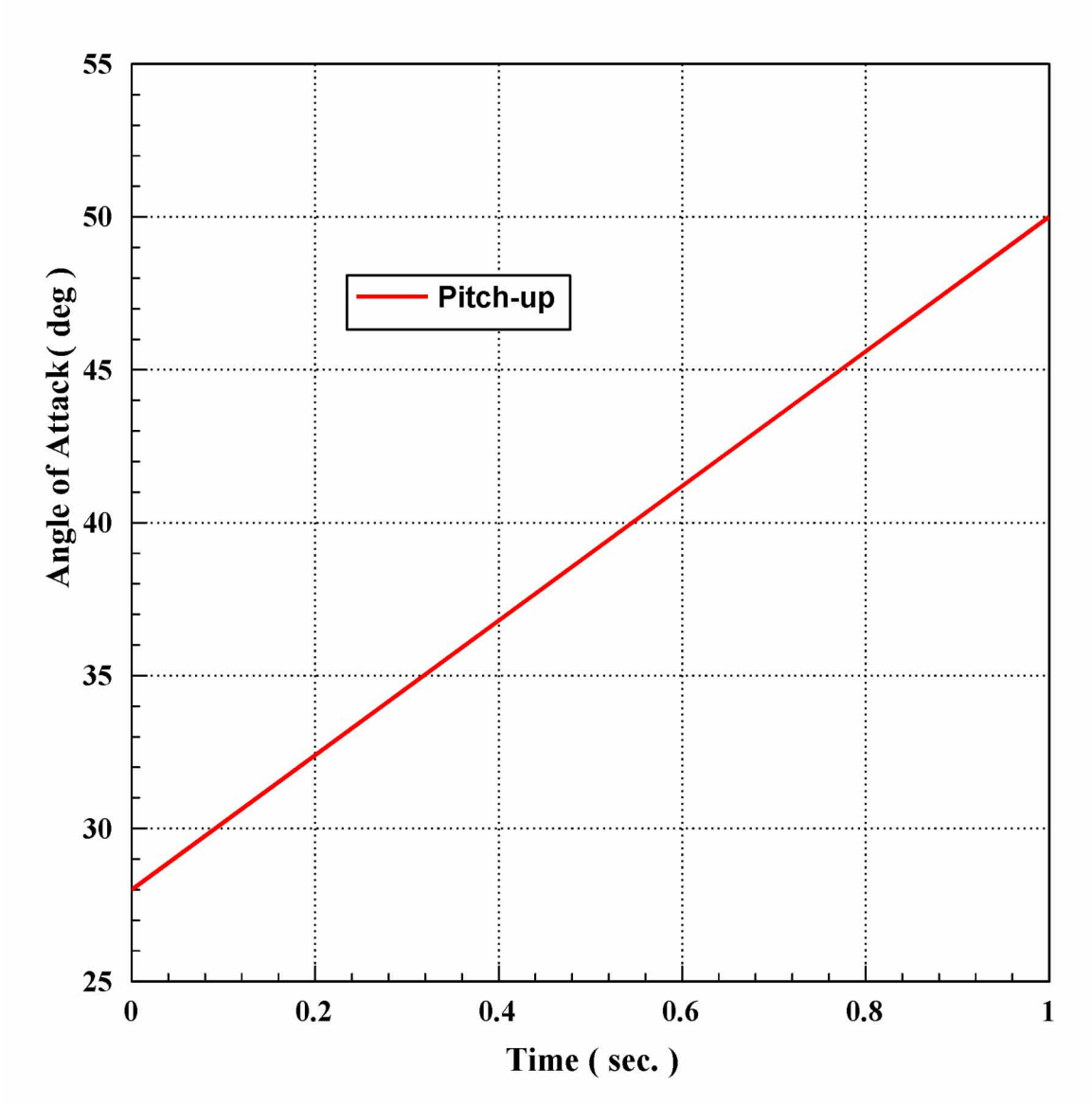
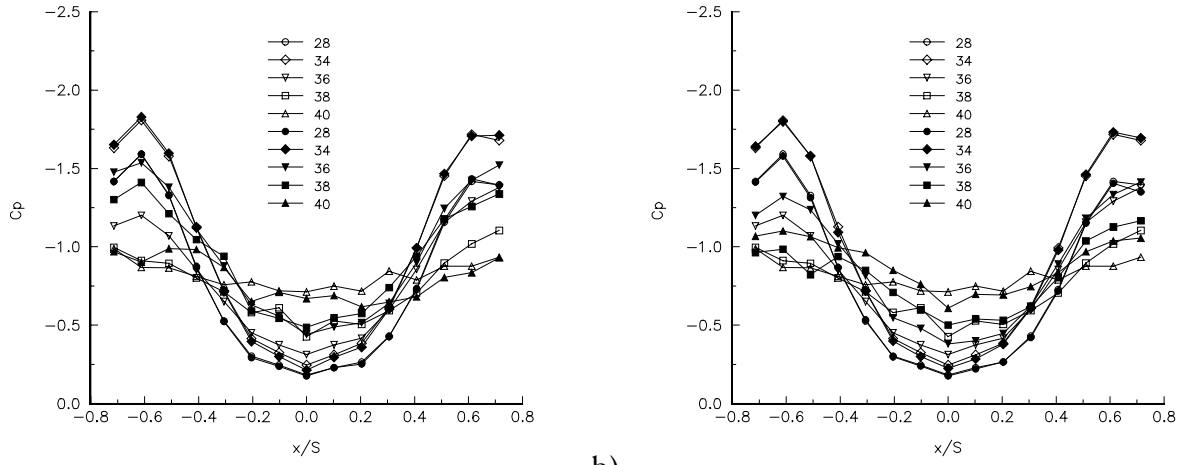


Figure 5.3: The motion schedule used in this phase of the research effort.

5.1.3 Results

The time history of the surface pressure distribution on two crossflow planes was obtained for the White model undergoing a pitch-up maneuver. Two-hundred time records were obtained during the maneuvers. As with the other experiments, during all of the pitch-up maneuvers, the model rotates about an axis through the apex of the model. The distribution of the pressure taps on the two crossflow planes is represented as follows. Again, the non-dimensional distance of a given port off the centerline of the model is termed x/S. The distance x is defined as the distance along a line perpendicular to the leading edge (Figure 2.8), i.e. the local half span, and S is the total length of the local half span. Therefore, x/S has values that run from ± 1, where -1 is the leading edge on the left side of the wing and 1 is the leading edge on the right side of the wing.

In Figure 5.4, the pressure distribution on the aft crossflow plane (y/C = 0.80) is shown for five angles of attack for both the baseline, i.e. no flap deployment, case and the 30° flap deployment case. The solid symbols correspond to the 30° flap deployment case and the empty symbols correspond to the baseline case. Analysis of the baseline data indicates that breakdown occurs over the aft crossflow plane at an angle of attack of 34°. For the steady case, breakdown is observed at this station at 30°. This indicates that the dynamic lag is small in this research effort, due to the low reduced frequency utilized. This, however, does not affect the ability to judge the effects of the apex flap. The effect of the apex flap is very strongly defined especially at $\alpha =$ 36° and 38°. Taking into consideration the time needed to deploy the flap once the trigger is sent, as discussed earlier, the apex flap is fully deployed in the 30° case by α = 31.75°. However, the effect of the flap deployment is not felt until α =36°. In fact, the curves for α=34° are identical for the two cases. This indicates that there is an appreciable lag in the response of the flow to the flap deployment.



a) b)

Figure 5.4: Distribution of the pressure coefficient on the Aft line of pressure taps for selected angles of attack. a) 30 flap deployment vs. baseline, b) 35 flap deployment vs. baseline. Lines plotted with empty symbols are for the baseline case. Lines plotted with solid symbols are for the cases with flap deployment.

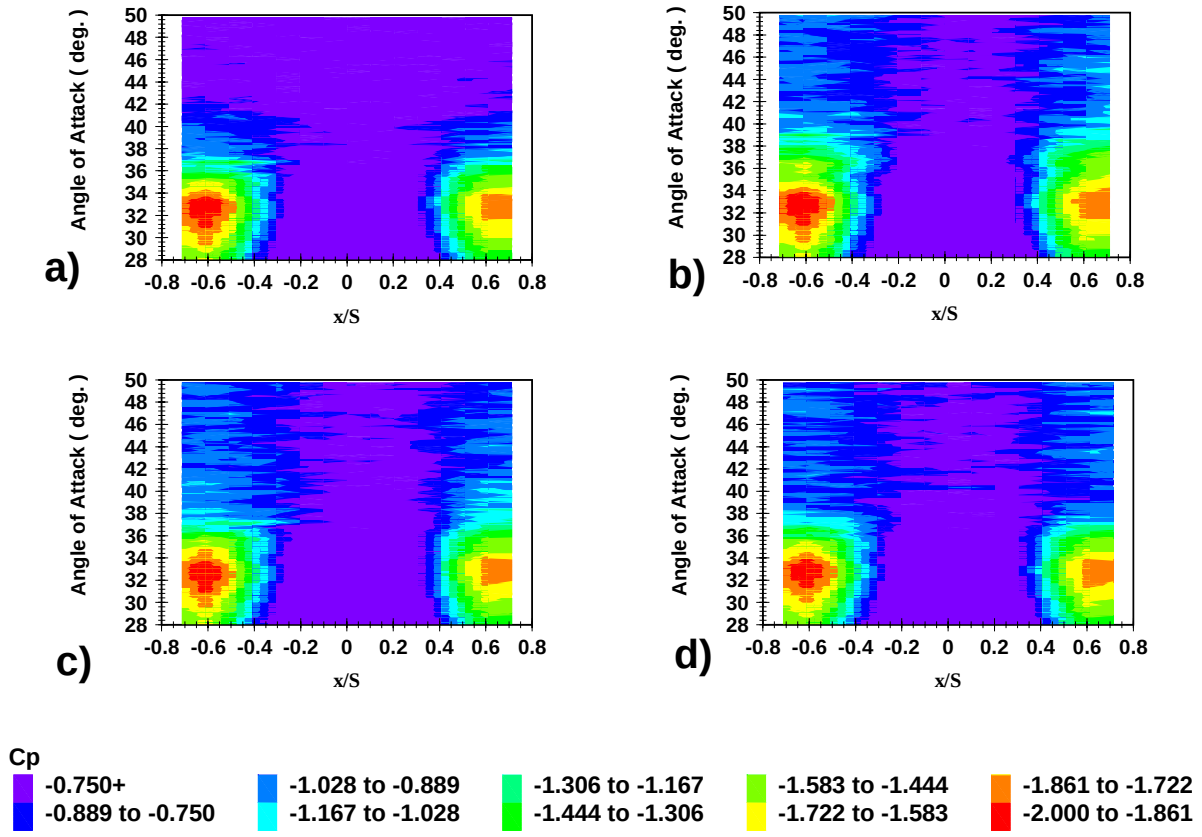


Figure 5.5: Time record of the variation of the pressure coefficient on the Aft line of pressure taps, y/C = 0.80, for a pitch-up maneuver. k=0.0108. Re=1,000,000 a) Baseline (no flap deployment), (b) 30° flap deployment, (c) 35° flap deployment, (d) 40° flap deployment.

In Figure 5.4, only a small number of the 200 pressure distributions available could practically be displayed. In Figures 5.5 and 5.6, a more complete presentation of the data is possible. In these figures, the data are presented with x/S values on the abscissa and the angle of attack on the ordinate. Since the pitch-up is a ramp function, angle of attack and time can be used interchangeably. The pressure coefficients are represented by different colors (shades) as noted in the legend. This form of data presentation allows the complete

time history to be displayed, as opposed to just presenting data from selected angles of attack.

Figure 5.5 shows the pressure distribution on the aft line of pressure taps (y/C = 0.80) for the baseline and the 30°, 35°, and 40° flap deployment cases. Figure 5.6 shows data for the same cases on the fore line of pressure taps (y/C = 0.70). The data presented within this figure shows certain features that are significant. In all cases, deployment of the apex

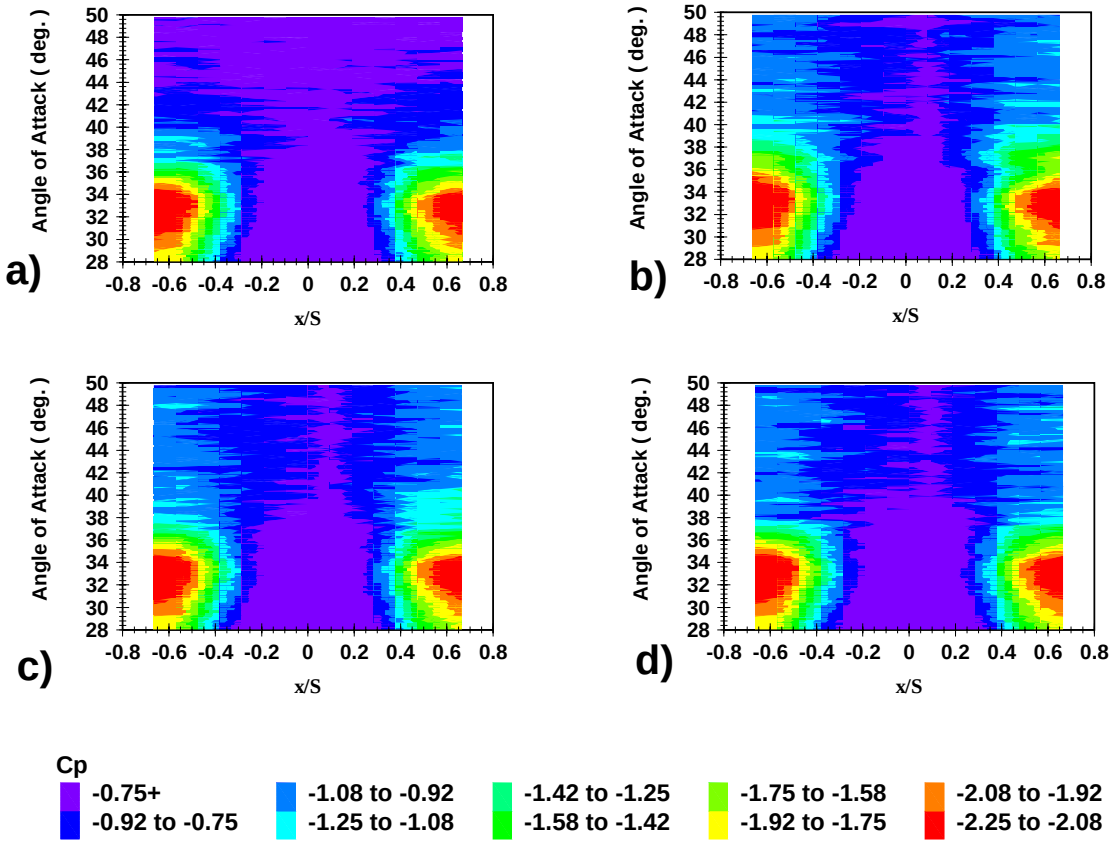


Figure 5.6: Time record of the variation of the pressure coefficient on the Fore line of pressure taps (y/C = 0.70), for a pitch-up maneuver. k = 0.0108. Re = 1,000,000. a) Baseline (no flap deployment), b) 30° flap deployment, c) 35° flap deployment, d) 40° flap deployment.

flap is not able to move the angle of attack at which the maximum suction occurs to a position later in the schedule. In the 30° flap deployment case, a secondary suction peak is seen to form in the range α=36°-38° although it is not as strong as the primary suction peak. This is the same as seen in Figure 5.4. In the 30°, 35°, and 40° cases, deploying the apex flap significantly changes the post-stall pressure distribution. In each case, the residual pressure distribution has a larger negative magnitude. This could be exploited as a possible means of post-stall control.

In Figure 5.7, we present the pressure coefficient as a function of angle of attack. Only the pressure coefficients seen at the port with the highest suction on each of the crossflow planes is plotted. For the aft line this is the first port. For the fore line, this is the second port. It can be seen that the peak in the pressure distribution occurs at the same angle of attack, regardless of flap deployment. The secondary suction peaks achieved by deploying the flap at 30° can also be seen. In each case, deploying the flap increases the amount of suction present over the baseline case.

5.2 Apex Flap Deployment during a 20° to 50° Pitchup

5.2.1 Experimental Conditions

The experimental conditions are the same for this set of runs as it is for the previous set with one exception, the way in which the flaps are triggered. During the dynamic motion of the model, one of the outputs of the DyPPiR control computer outputs a clock train in sync with the digital-to-analog conversions for the DyPPiR control signals. This clock train is routed to a counter chip on a ComputerBoards, Incorporated DAS-08, which has been programmed with a terminal count. When the terminal count has been reached, a trigger is sent to the control valve assembly and the flap is deployed. Experiments have shown that there is a 0.135 second delay between the trigger and full deployment of the flap. The trigger time was adjusted so that the flap deployed at the proper time in the pitch-up schedule. All

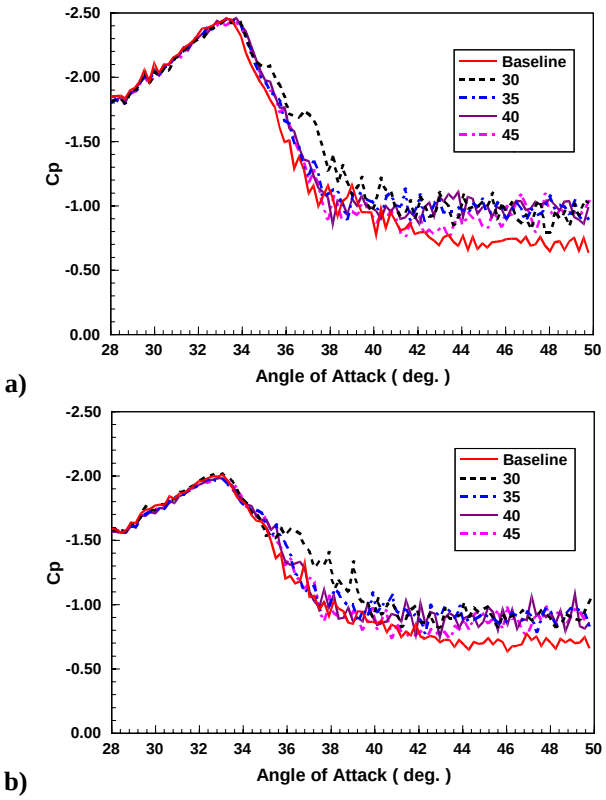


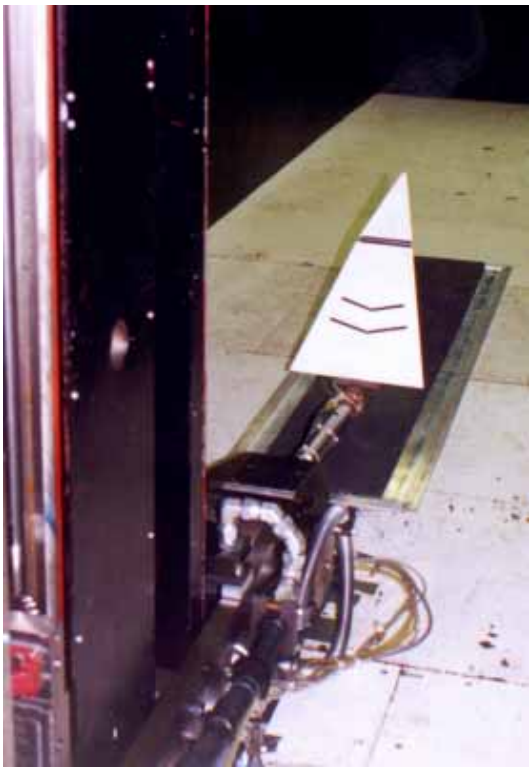
Figure 5.7: Pressure coefficient as a function of angle of attack for the port experiencing the maximum pressure on the line a) Fore line – Port 1 b) Aft line -- Port 2.

deployment angles reported herein are the actual time, or angle of attack, of the flap deployment. Figure 5.8 shows the White wing mounted on the DyPPiR for this set of runs.

5.2.2 Description of Motions

The two pitch-up schedules used in this set of experiments are shown in Figure 5.9. Each consists of a ramp function from 20° to 50° angle of attack in 1.3636 and 0.5 seconds, respectfully. The motions are named as follows: Motion 1 is the 1.3636 second ramp, and Motion 2 is the 0.5 second ramp. In all of the pitch-up maneuvers, as always, the model pitches about an axis through the apex of the model.

a)



b)

Figure 5.8: The model mounted on the DyPPiR in the Virginia Tech Stability Wind Tunnel, (a) from within the tunnel, (b) from outside the tunnel. The valve block which controls the flap deployment can be seen in the lower left hand corner of (b).

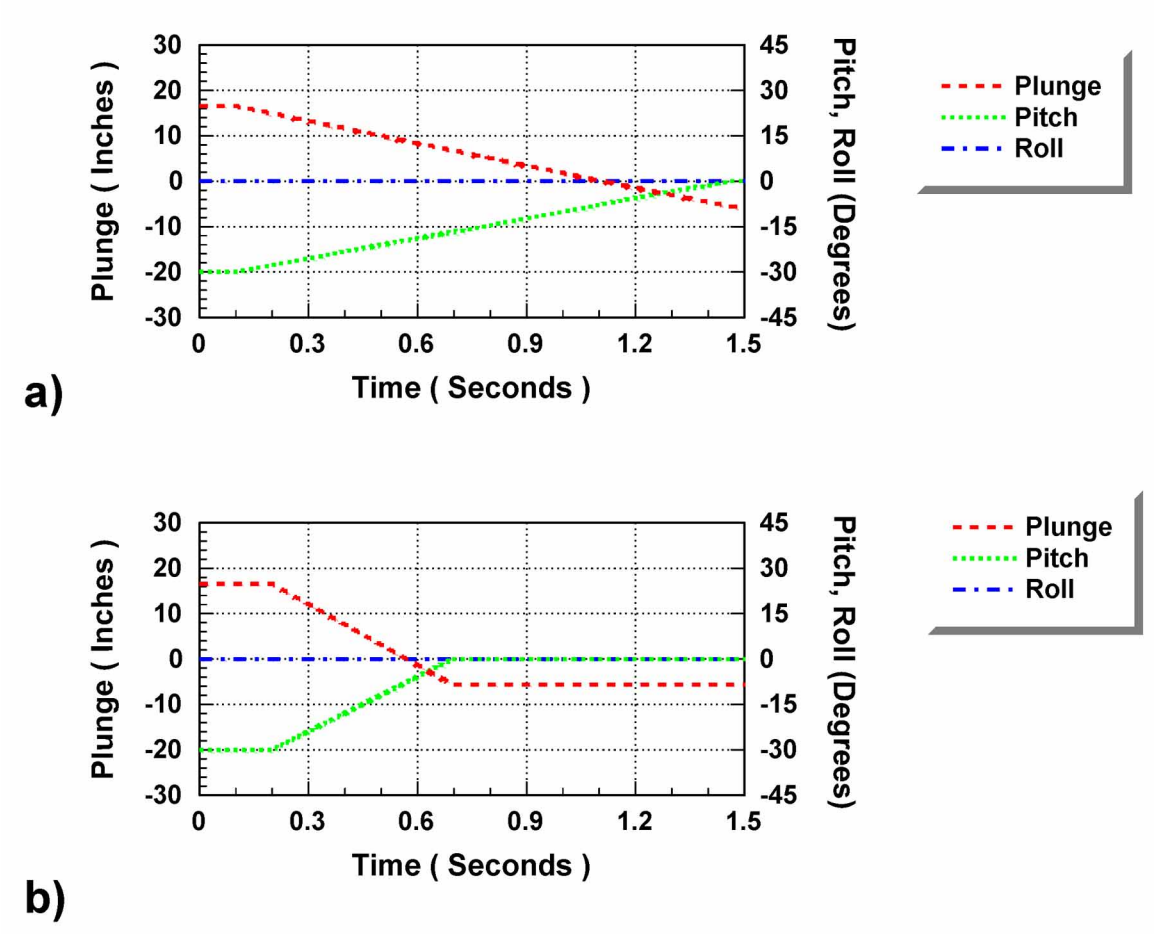


Figure 5.9: The motion schedules utilized in the current research. All the motions represent a ramp-like pitch-up from 20° to 50° in (a) 1.3636 seconds (Motion 1) and (b) 0.5 seconds (Motion 2). To see Motion 1 as seen in the DyPPiR Simulator for a flap deployment of 32°, click on (a) above.

The model stays at the maximum angle of attack for 5 seconds after the motion is complete. Each motion is 1.5 seconds in length. The DyPPiR pauses twenty seconds between returning the model to its original position and the beginning of the next realization of the motion. For each experimental run, 30 realizations of the motion were performed, then ensemble averaged.

5.2.3 Results and Discussion

The same data acquisition routines and data representations used in the previous section are used here. The data presented in Figure 5.10 is for the pressure taps on the Fore line (y/C=0.70) during a Motion 1 pitch-up maneuver at a Re of 1,000,000 and a k of 0.0053. The data show that deploying the flap can have very beneficial effects. When the flap is deployed right at the start of the motion, at an angle of attack of 21°, there is an increase in both the duration and the strength of the suction peak (Fig. 5.10b) over the baseline case (Fig. 5.10a). There is also a positive effect if the flap is deployed at 28° (Fig. 5.10c). By plotting the pressure coefficient as a function of time for the pressure tap under the suction peak, we can gain insight into exactly the effect of the flap deployment. These data are presented in Figure 5.11. Here it is quite clear the effect of the flaps deploying. The suction peak for the 21° flap deployment case is shifted to an angle of attack 3.5° higher than the baseline case and the strength of the suction peak is 15% greater than in the baseline case. Deploying the flap at 28° angle of attack results in a 1° shift in angle of attack and an increase in strength of 9%. Another feature that can be seen in both Figures 5.10 and 5.11 is that deploying the flap raises the pressure coefficients in the post-stall region over the value of the baseline case. Indicating that overall, the suction on this section with the flap deployed will be greater than if the flap was not deployed. In Figure 5.12, data are presented for the pressure taps on the Fore line during a Motion 2 pitch-up maneuver at a Re of 1,000,000 and a k of 0.0144. In contrast to the data presented in Figure 5.10, the data in Figure 5.12 are for a higher pitch rate, and thus a higher k. But the Reynolds number is the same in both cases, so the freestream velocity has remained the same. Also since the rate at which the flap is deployed was not adjustable in this set of experiments, the reduced frequency of the flap deployment will be the same in both cases. This appears to have an effect on the ability of the flap to increase the location and the strength of the suction peak. The data for the 20° deployment shows an increase in strength of only 9% and a shift in angle of attack of only 1°.

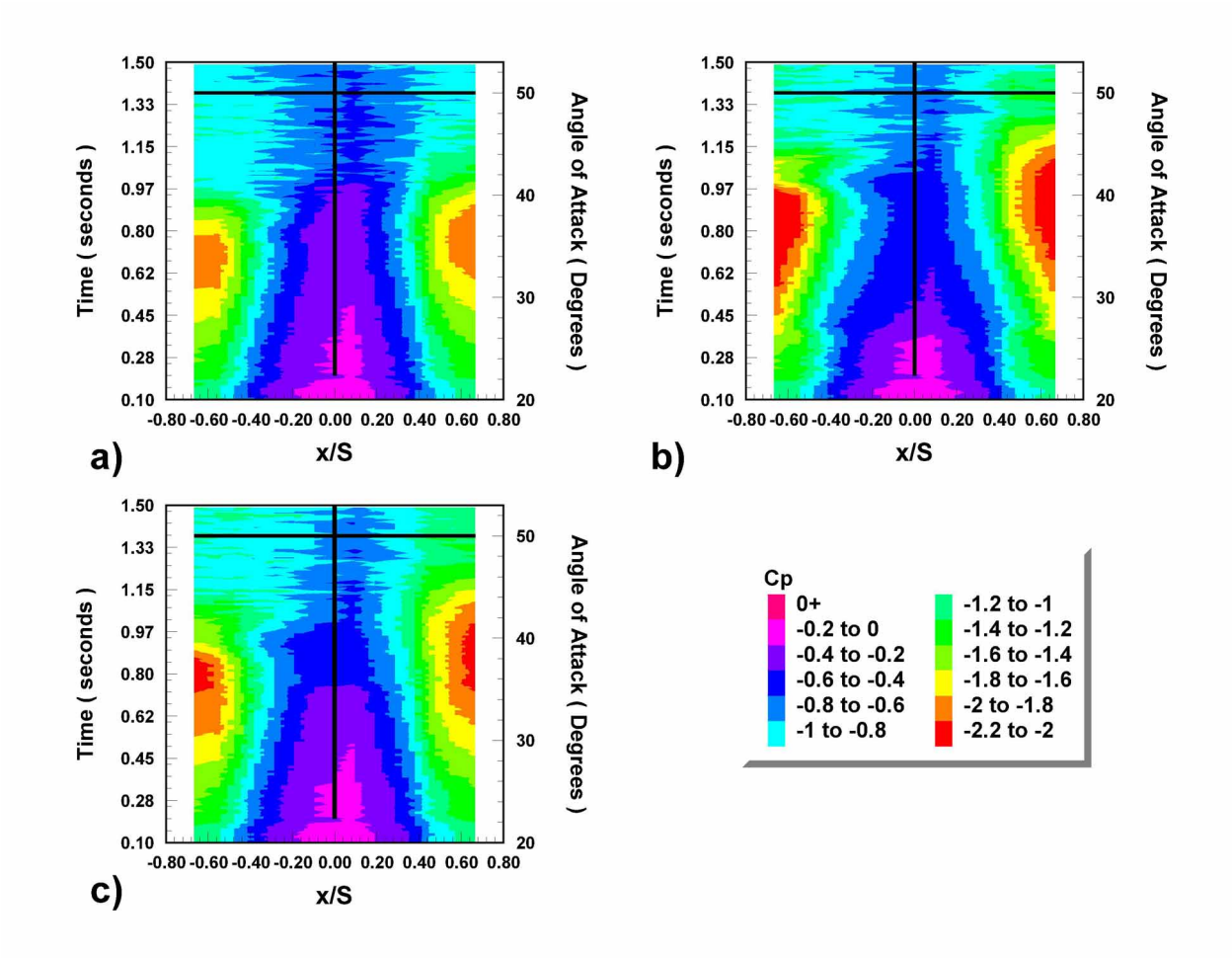


Figure 5.10: Time record of the variation of the pressure coefficients on the Fore line of pressure taps (y/C=0.70), for the Motion 1 pitch-up maneuver. k = 0.0053, Re=1,000,000. a) Baseline, b) 21° flap deployment, c) 28° flap deployment. The vertical line represents the centerline of the wing.

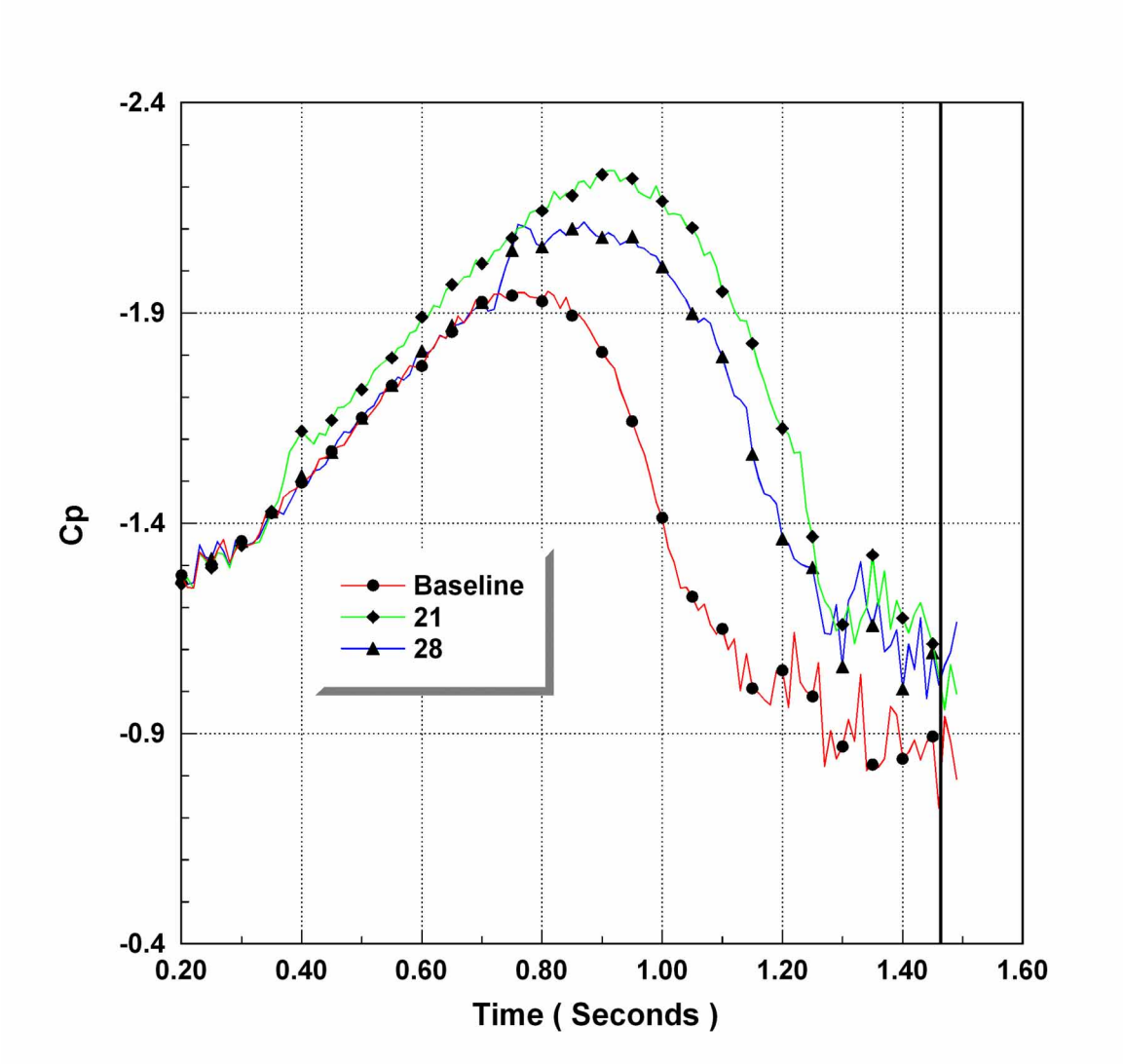


Figure 5.11: Pressure coefficient as a function of time for the port on the Fore line that experiences the maximum surface pressure for the Motion 1 pitch-up maneuver. Re = 1,000,000 and k = 0.0053. The vertical line indicates the time at which the motion stopped.

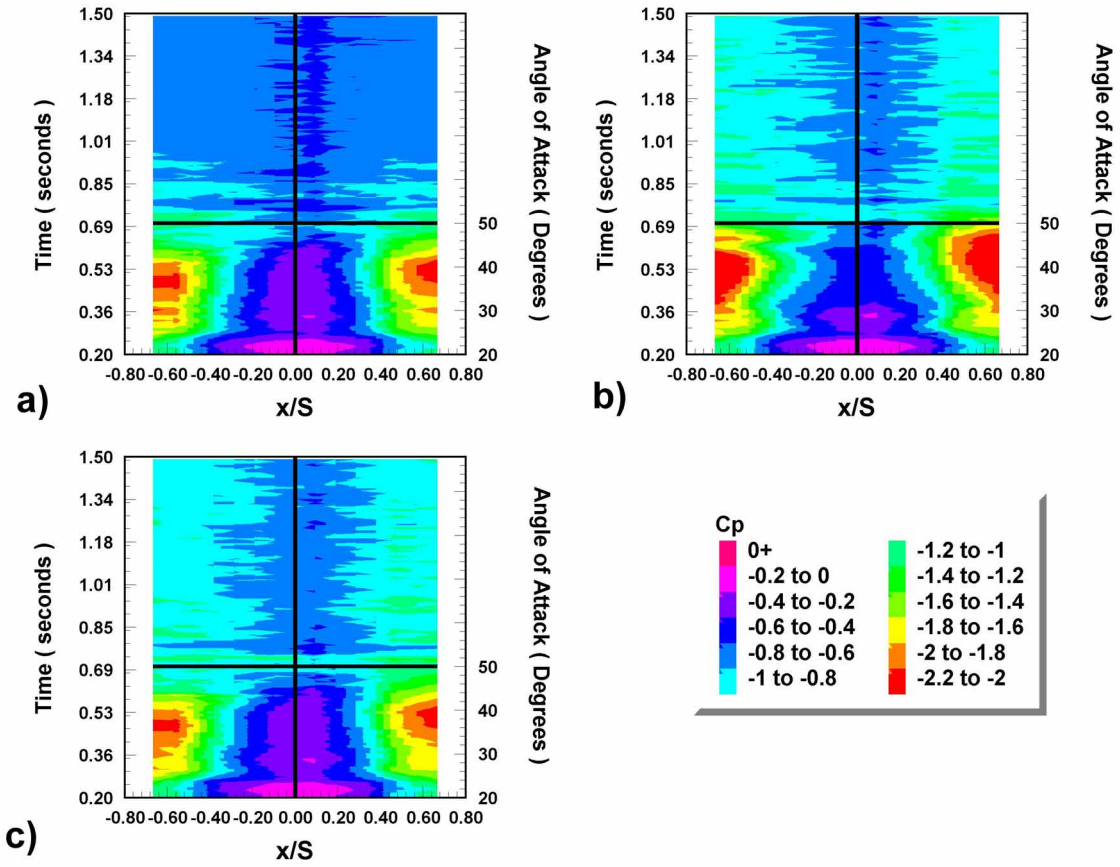


Figure 5.12: Time record of the variation of the pressure coefficients on the Fore line of pressure taps (y/C=0.70), for the Motion 2 pitch-up maneuver. k = 0.0144, Re=1,000,000.a) No flap deployment, b) 20° flap deployment, and c) 30° flap deployment. The vertical line represents the centerline of the wing and the horizontal line indicates when the motion stopped

Deploying the flap at 30° yields a surface pressure distribution almost identical to the baseline case, with the exception of the post-stall region. In that region the pressure coefficients move from resembling the baseline case to resembling the 20° deployment case. This can be clearly seen in the time record of the pressure coefficients presented in Figure 5.13. It remains to be seen if the differences between the data presented in Figures 5.10 and 5.12 can be attributed to a flap reduced frequency effect. The baseline time record from Figure 5.13 is shown in Figure 5.14 along with the distribution of the non-dimensional RMS of the coefficient of pressure with time. The RMS is defined here as

$$\frac{RMS(C_P')}{\overline{C_P}} = \frac{\sqrt{\frac{1}{N}\sum (C_P - \overline{C_P})^2}}{\overline{C_P}}$$

The non-dimensional RMS distribution presented here is representative for the motions described herein. Once breakdown has begun to sweep over the surface of the wing, as indicated by the decline in the coefficient of pressure, the RMS values dramatically rise and fluctuate. Similar behavior has been seen for static and dynamic motions of delta wings by Rediniotis (1992) and Gursul and Yang(1994).

The data presented in Figure 5.15 is also for a Motion 2 pitch-up maneuver at a Re of 1,000,000 and a k of 0.0144. Here the data is for the Aft line of taps (y/C=0.80). A full compliment of flap deployment angles are presented from 20° to 35°. The same general trend is indicated, the most benefit is extracted from the flap being deployed when that event occurs as soon as possible in the motion schedule. Special attention should be called to Fig. 5.15f. The data presented there are for the flap being deployed throughout the motion, i.e. the flap is always deployed. It can be seen by comparing 5.15f and 5.15e and 5.15d, that it is better to have the flap always deployed than to deploy it late in the schedule. This is due to the same type of time lag that postpones breakdown to a higher angle of attack during a

dynamic motion. Any effect associated with changes to the flow field near the apex take time to propagate downstream. By the time the effect of a late flap deployment is felt, the vortex

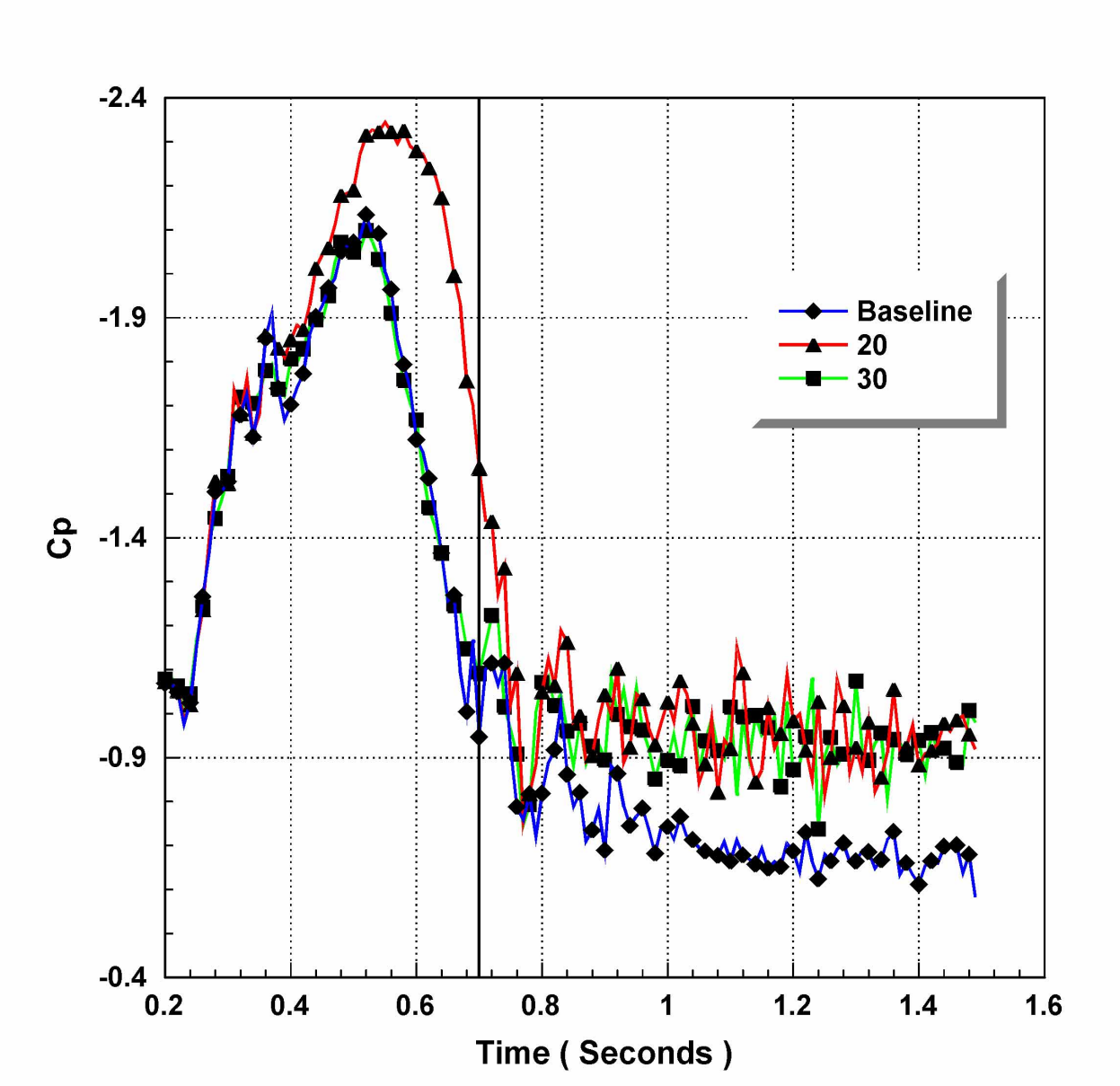


Figure 5.13: Pressure coefficient as a function of time for the port on the Fore line that experiences the maximum surface pressure for the Motion 2 pitch-up maneuver. Re = 1,000,000 and k = 0.0114. The vertical line indicates when the motion stopped.

is already too close to breaking down to have an effect. An interesting comparison can be made of 5.13f and 5.13b. Here the effect of deploying the flap at α=20° is greater than the effect of having the flap deployed continuously. This would seem to indicate that the motion of the flap, and the corresponding vorticity that the motion produces, is an important effect. The flap deployment adds an additional effect above what is present due to holding the apex at a lower angle of attack prior to the maneuver.

Additional evidence that the reduced frequency of the flap deployment is a critical parameter can be seen in Figure 5.16. Here time has been non-dimensionalized based on the time of the pitch-up motion, 0.0 is the start of the motion and 1.0 is the end of the motion. For these two runs the reduced frequency of the pitch-up motions is the same, k = 0.011, but the Reynolds numbers are different. For Figure 5.16a, the schedule is Motion 1 and Re = 532,000. For Figure 5.16b, the schedule is Motion 2 and the Re = 1,000,000. In both cases the flap is deployed at about 25° angle of attack, 23° and 25°, respectfully for Motion 1 and Motion 2. Again the time to deploy the flap was not adjustable in this experiment, so if we create a reduced frequency for the flap motion based on the time it takes to deploy the flap, that reduced frequency would be, for the data in Figure 5.16a, twice that for the data in Figure 5.16b. The effect of the flap deployment is significantly more pronounced in Figure 5.16a than in 5.16b. The Motion 1 data has a suction peak with a pressure coefficient over –2.0 while the Motion 2 data has pressure coefficient peak only on the order of –1.7.

Since the motions of the White wing on the DyPPiR were video taped is a straightforward task to digitize the video tape and arrive at an approximate time to deploy the flaps. In Figure 5.17, four such frames are presented. Analysis of other video taped sequences reveal that what is shown here is typical. In the frames of Figure 5.17, we see an approximate deployment time of 0.05 to 0.07 seconds. Digital video of this pitch-up can be seen in Media Object 5.1.

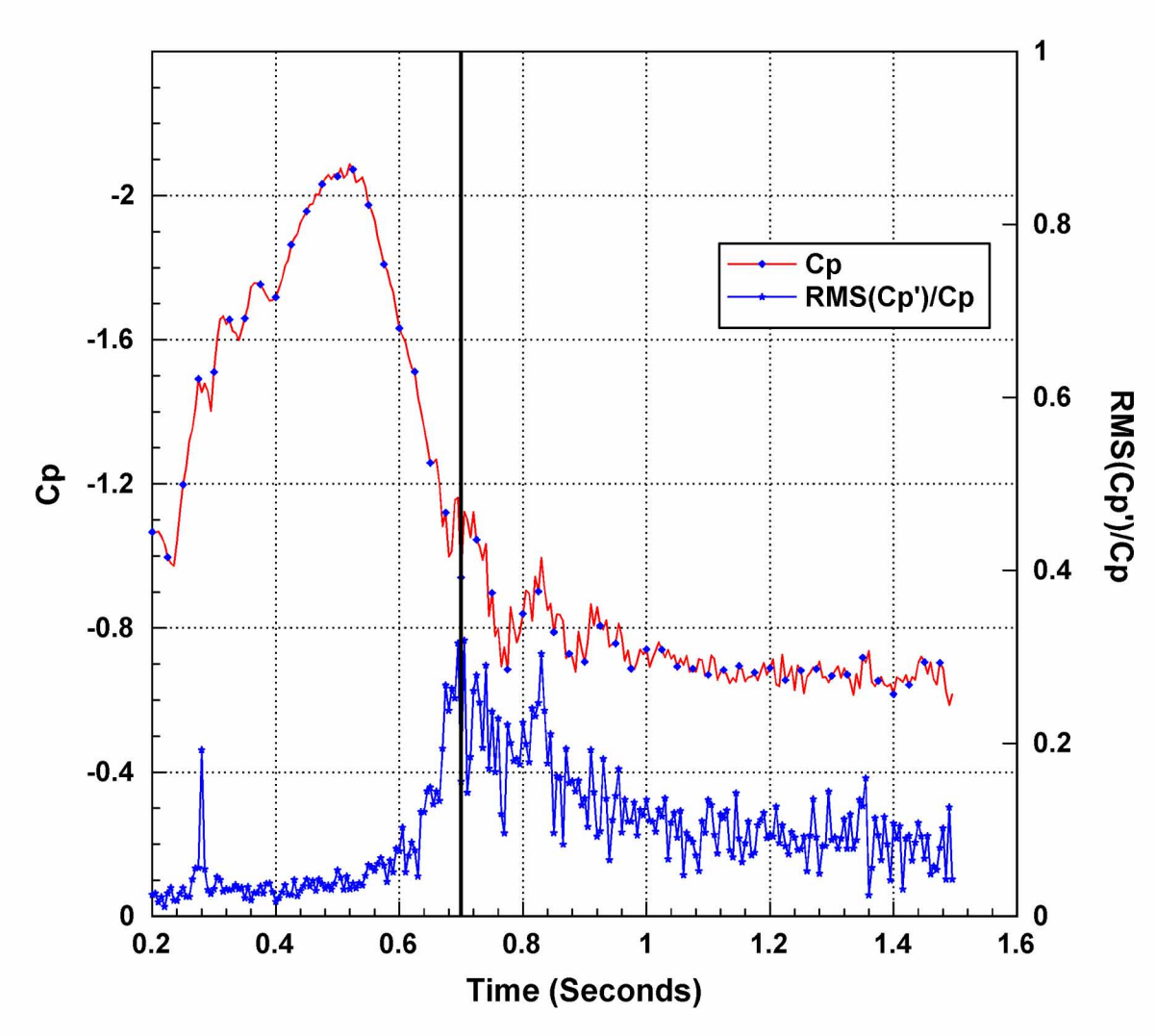


Figure 5.14: Coefficient of pressure and the RMS of the coefficient of pressure as a function of time for the port on the Fore line that experiences the maximum surface pressure for the Motion 2 pitch-up maneuver. Re = 1,000,000 and k = 0.0114. The vertical line indicates when the motion stopped.

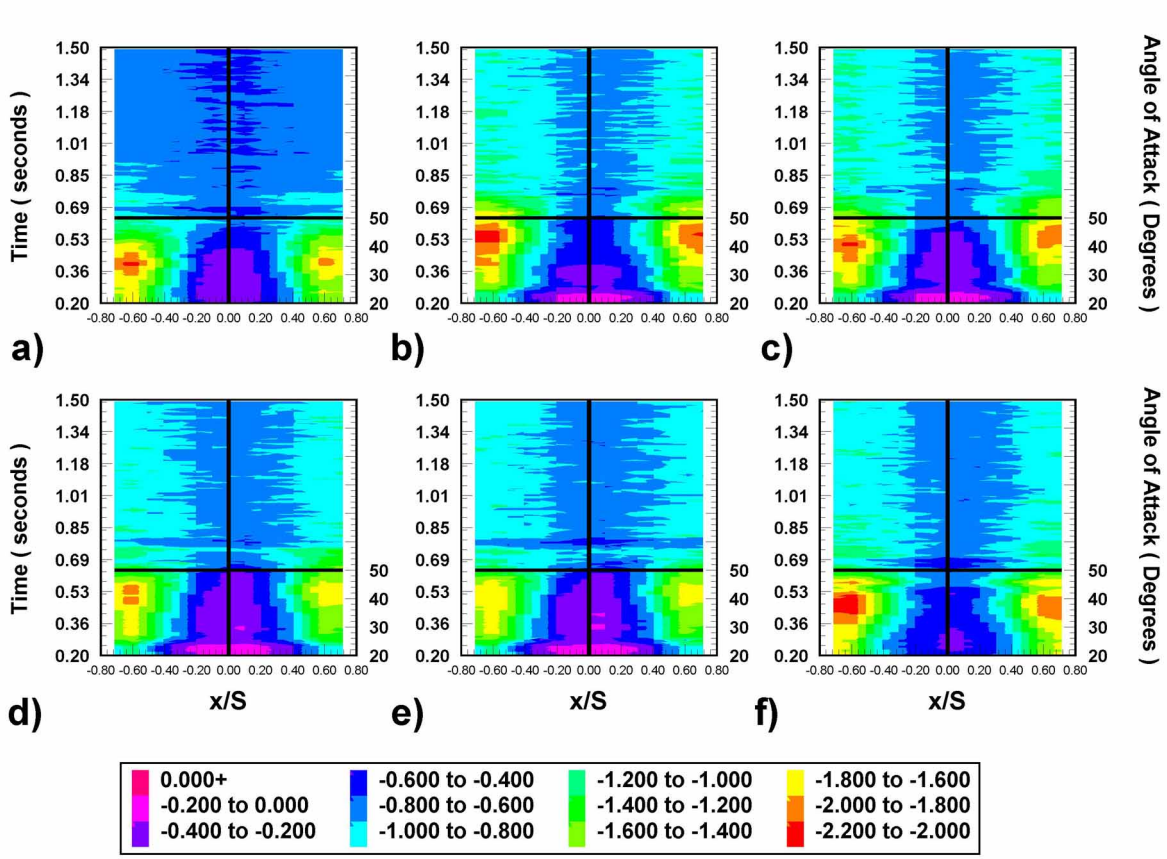


Figure 5.15: Time record of the variation of the pressure coefficient on the Aft line of pressure taps (y/C = 0.80), for the Motion 2 pitch-up maneuver. k = 0.0144 Re = 1,000,000 a) No flap deployment, b) 20° flap deployment, c) 25° flap deployment, d) 30° flap deployment, e) 35° flap deployment, and f) flap always deployed. The vertical line represents the centerline of the wing and the horizontal line indicates when the motion stopped.

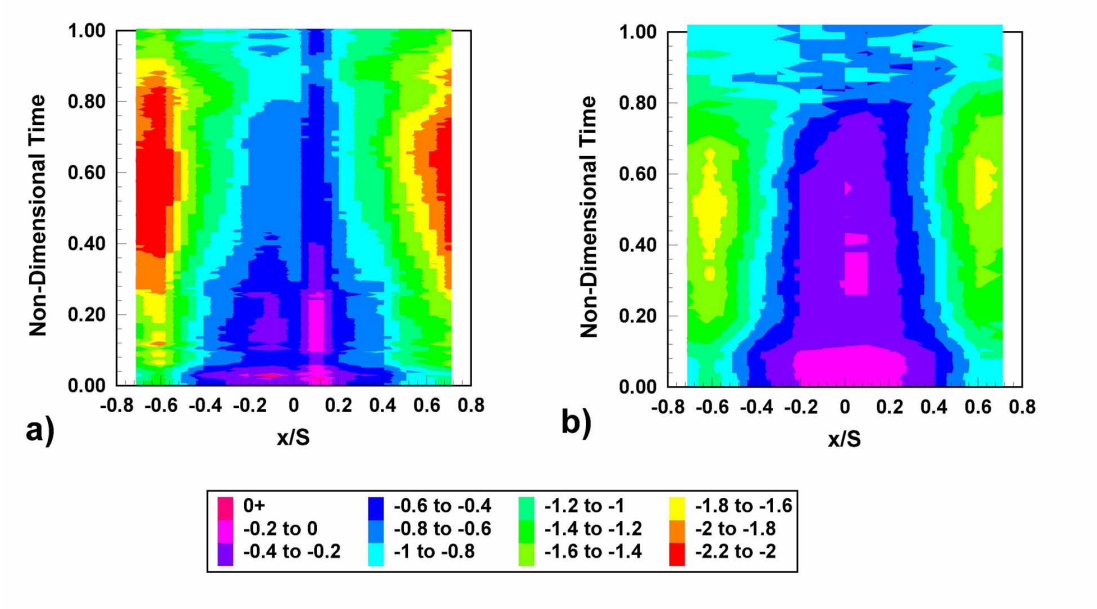


Figure 5.16: Surface pressures for k = 0.011 a) Motion 1, 23° flap deployment, Re=533,000, b) Motion 2, 25° flap deployment, Re=1,390,000.

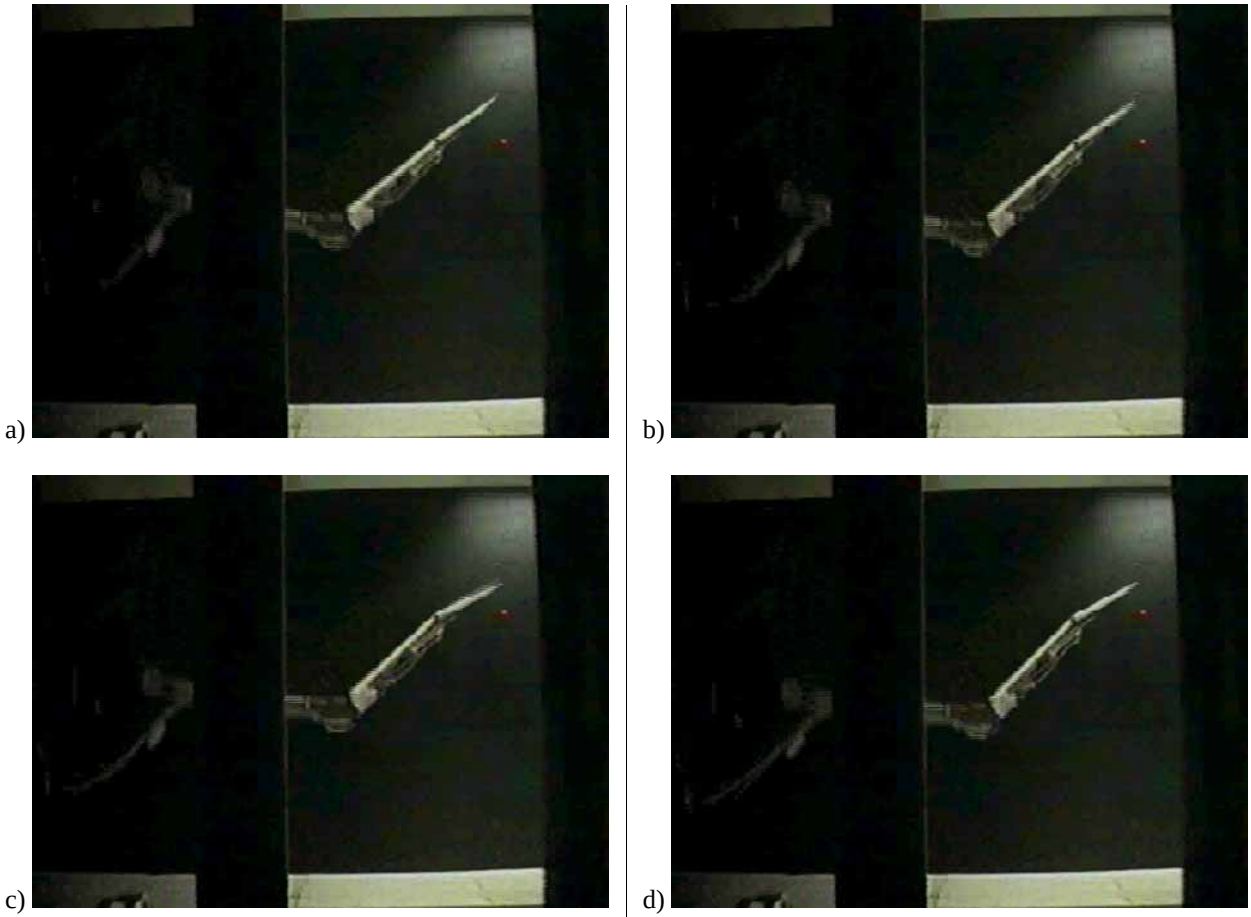


Figure 5.17: A sequence of captured video frames from a standard videotape of the apex flap deployment. Frame (a) appears to be before the deployment has begun, (b) and (c) show the flap in the process of deploying, by (d) the flap is fully deployed. The frames are sequential indicating it takes between 0.05 and 0.07 seconds for the flap to deploy.

Media Object 5.1: Digital video sequence of the pitch-up motion with apex flap deployment. Video sequence is the same one from which the above frames (Figure 5.17) were captured. Click on above image to see the video.

CHAPTER 6: SUMMARY, CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

The flowfield over a 75°-sweep angle delta wing was investigated using two non-intrusive measurement techniques. Velocity fields, one-dimensional and three-dimensional, were acquired using laser-Doppler velocimetry (LDV). Surface pressure measurements were made using an electronic pressure scanner. LDV measurements were made over a delta wing at a fixed angle of attack, in either a plain configuration or equipped with cavity flaps. The surface pressure measurements were conducted on a delta wing at fixed angle of attack with and without cavity flaps, over a delta wing executing a pitch-up maneuver with flaps continuously deployed and where the flaps were deployed during the motion. The dynamic motions were conducted for a wing with cavity flaps and a wing with an apex flap. The following data matrix summaries all the data presented herein:

Condition	Angle of Attack(s)	Re	Measurements	Comments
Steady	38°	72,000	Velocities	3-D Velocity Field and frequency analysis within the field.
Steady	35°	75,000	Velocities	Axial Velocities With and Without Cavity Flaps
Steady	$28° \le \alpha \le 44°$ 2° increments	1,000,000	Surface Pressures	With and Without Cavity Flaps
Steady	$22° \le \alpha \le 40°$ 2° increments	470,000	Flow Visualizations	With and Without Cavity Flaps
Unsteady k=0.0089	Pitchup from 28° to 42°	1,300,000	Surface Pressures	With and Without Cavity Flaps
Unsteady k=0.0089	Pitchup from 22° to 42°	1,300,000	Surface Pressures	Cavity Flaps deployed at pre-set angle during motion.
Unsteady k=0.0089	Pitchup from 28° to 60°	187,000	Surface Pressures	Cavity Flaps deployed at pre-set angle during motion.
Unsteady	Pitchup from 28° to 50°	1,200,00	Surface Pressures	Plain wing: k=0.0089, 0.0125, 0.0161, 0.0197, 0.0233, and 0.0267.

Unsteady k=0.0108	Pitchup from 28° to 50°	1,000,00	Surface Pressures	Apex Flap Deployment at α=30° and α=36° and no flap deployment.
Unsteady k=0.0053	Pitchup from 20° to 50°	1,000,000	Surface Pressures	Apex Flap Deployment: Never, α=21°, α=28°
Unsteady k=0.0144	Pitchup from 20° to 50°	1,000,000	Surface Pressures	Apex Flap Deployment: Never, α=20°,25°,30°,35°, and always.
Unsteady k=0.0114	Pitchup from 20° to 50°	532,000	Surface Pressures	Apex flap Deployment: α=23°
Unsteady k=0.0114	Pitchup from 20° to 50°	1,390,000	Surface Pressures	Apex Flap Deployment: α=25°

6.1 Conclusions

From these experimental observations, the following conclusions can be drawn.

1. From the detailed three-dimensional velocity field over the delta wing at an angle of attack of 38°, the sectional streamline pattern upstream of breakdown features an unstable focus, while downstream of breakdown features an unstable focus, while downstream of breakdown it exhibits two limit cycles, a repelling and an attracting one. The unstable character of the saddle-to-saddle connection on the symmetry plane of the wing, was experimentally documented. The behavior of the 3-D streamlines was investigated in the region of the breakdown stagnation point, which is a nodal critical point. It was observed that in the neighborhood of the vortex core, the vortex lines and the streamlines are aligned and coincident with one another upstream of breakdown. Shortly upstream of breakdown the vortex lines abruptly diverge from the streamlines.

   The following table shows when each indicator of vortex breakdown indicated that breakdown had begun.

Indicator	Occurrence (X/C')
Axial Velocity	0.744
Normalized Helicity	0.742
Unstable Focus	0.592

2. The experimental data presented here indicate that cavity flaps can delay considerably vortex breakdown in steady flow. In the range of angles of attack 28° to 42°, for each fixed angle, flaps can displace vortex breakdown downstream by about a third to a half of a chord length. For unsteady flow, the situation is quite different, at least for the range of angles of attack tested here, $28°<\alpha<42°$. The dynamic maneuver is allowing the leading edge of the wing to generate additional vorticity, which feeds directly into the vortex and sustains its coherence and introducing a lag in when vortex breakdown will occur over the wing. With a dynamic motion terminating at $\alpha=42°$ and the flaps continuously deployed, our pressure data indicate that right at the end of the motion to pressure seizes to drop and reverses direction. The effect of cavity flaps in the dynamic case is not as pronounced as in steady flow. Indeed it appears that the minimum in the low pressure over the plain wing is reached at about $\alpha=40°$. On the other hand, cavity flaps appear to delay the appearance of the minimum and essentially shift the behavior to higher angles of attack. Cavity flaps should therefore offer a stabilizing effect to the vortical structures. In fact our results indicate that in dynamic motions, it would be more efficient to deploy the cavity flaps during the motion, but in this case, they need to be deployed as close to the beginning of the motion as possible.

3. The apex flap was evaluated as a delta wing control surface, before and after the effects of vortex breakdown were felt on the wing. The deployment time of the apex flap during the motion was varied and it was found that if the flap was deployed before breakdown occurred over the wing, there was a beneficial effect. Additional suction peaks were observed in the pressure distribution, although these peaks were not as large as the pre-

breakdown suction peak. Changes in the surface pressure peaks, as large as 15%, over a baseline no flap deployment case, were observed. The motion of the flap as it deploys was found to have an additional effect on the changes in the flowfield. A flap deploying during the motions will have a more pronounced effect than if the flap is deployed continuously. This could possibly be due to the added vorticity associated with the movement of the flap. If the apex flap is deployed later in the schedule, after breakdown is over the planform of the wing, the effect is not as strong as if the flap is deployed before. But it is quite clear that the apex flap does have a positive effect in the post-stall regime.

The most general conclusion that can be drawn from this body of work is that once the vortex is broken down there is very little a control surface can do to bring it back. If you wish to re-energize the core of the vortex, you must put that "energy" into the vortex before breakdown. An ounce of prevention is once again worth a pound of cure. And just like that famous egg, once the vortex is broken down, even "All the King's horses and all the King's men" will not be able to put it back together again.

6.2 Recommendations for Future Work

Based on the results of this investigation, the author feels that the following would be interesting areas of future research and would tend to broaden the results presented herein and the general state of knowledge of delta wing aerodynamics.

1. Velocity Field Near the Apex: Since it has been shown here that, at high angle of attack, only the part of the leading-edge forward of 35% of the projected chord sheds vorticity that finds its way into the core, it would be interesting to study more carefully the details of the flow in that region. How much of the leading-edge flow actually feeds the core directly? In an apex-dominated flow such as this, what effect do the exact details of the apex geometry have on the overall flowfield.

2. Instantaneous Condition of the Saddle to Saddle Connection: Experimental evidence presented here shows that the saddle-to saddle connection, the separatrix, between the two vorticies over the wing is indeed unstable. What is the frequency of the separatrix breaking and is this frequency related to other frequencies observed in the flowfield and does the frequency change in the neighborhood of breakdown. Most computational studies are run using a symmetry plane. How does this phenomenon effect the results of these studies? This could be examined experimentally using high-speed digital particle image velocimetry. The author is currently involved in such an investigation.

3. Flap Deployment Reduced Frequency: What is the total effect of the flap deployment reduced frequency and how best to define such a non-dimensional parameter so that it could allow comparisons between flap systems.

4. Asymmetrical Cavity Flap Deployment: What is the effect of asymmetrical deployment of the cavity flaps? This investigation would be better served by force and moment measurements rather that surface pressure measurements. Could this be a valid rolling moment control at high angle of attack?

5. Flap Deployment During Composite Maneuvers: The effect of a flap deployment during a composite maneuver, such as the cobra maneuver, is a logical extension of the work presented herein and an excellent application of the DyPPiR to the study of unsteady aerodynamics.

These items would help to broaden the results of the work that has been presented here and would also greatly contribute to our understanding of delta wing aerodynamics.

REFERENCES

Aerospace and Ocean Engineering, Virginia Tech, 1997, Stability Tunnel Web Page, http://www.aoe.vt.edu/aoe/physical/stab.html

Agrawal, S., Barnett, R. M., Robinson, B. A., 1992, "Numerical Investigation of Vortex Breakdown on a Delta Wing," *AIAA Journal*, Vol. 30, No. 3, pp. 584-591.

Ahn, S., Choi, K.-Y., and Simpson, R. L., 1989, "The Design and Development of a Dynamic Plunge-Pitch-Roll Model Mount," AIAA Paper No-89-0048.

Ahn, S., 1992, "An Experimental Study of Flow Over a 6 to 1 Prolate Spheriod at Incidence," Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Department of Aerospace and Ocean Engineering, Blacksburg, Virginia.

Alcorn C. W., Croom, M. A., Francis, M. S., 1995, "The X-31 Experience: Aerodynamic Impediments to Post-Stall Agility," AIAA Paper No. 95-0362.

Alexander, M. G., 1993, "A Qualitative Assessment of Control Effects on an Advanced Fighter Configuration," AIAA Paper No. 93-3627.

Atta, R. and Rockwell, D., 1990, "Leading Edge Vortices Due to Low Reynolds Number Flow Past a Pitching Delta Wing," *AIAA Journal*, Vol. 28, pp. 995-1004.

Benjamin, T. B., 1962, "Theory of Vortex Breakdown Phenomena," *Journal of Fluid Mechanics*, Vol. 14.

Berdahl, C. H., and Thompson, D. S., 1993, "Eduction of Swirling Structure Using the Velocity Gradient Tensor," *AIAA Journal,* Vol. 31, No. 1, pp. 97-103.

Brown, G.L. and Lopez, J. M., 1990, "Axisymmetric Vortex Breakdown Part II: Physical Mechanisms," *Journal of Fluid Mechanics*, Vol 221, pp. 553-576.

Dallmann, U., 1983, "Topological Structures of Three-Dimensional Vortex Flow Separation," AIAA Paper 83-1735.

Delery, J. M., 1994, "Aspects of Vortex Breakdown," Progress in Aerospace Sciences, Vol 30, pp. 1-59.

Ericsson, L. E. , 1994, "Cobra Maneuver Unsteady Aerodynamics Considerations," *Journal of Aircraft*, Vol. 32, pp. 214-216.

Escudier, M., 1988, "Vortex Breakdown: Observations and Explanations," *Progress in Aerospace Sciences*, Vol. 25, pp. 189-229.

Faler, J. H. and Leibovich, S., 1978, "An Experimental Map of the Internal Structure of a Vortex Breakdown," *Journal of Fluid Mechanics*, Vol. 86, pp. 313-335.

Gad-el-Hak, M. and Blackwelder, R. F., 1985, "The Discrete Vortices from a Delta Wing," *AIAA Journal*, Vol. 23, No. 6.

Gursal, I. And Yang, H., 1994, "Vortex Breakdown over a Pitching Delta Wing", AIAA Paper No. 94-0536.

Gursul, I., Yang, H., and Deng, Q., 1995, "Control of Vortex Breakdown with Leading-Edge Devices," AIAA Paper No. 95-0676.

Helman, J. and Hesselink, L., 1989, "Analysis and Visualization of Flow Topology in Numerical Data Sets," *Topological Fluid Mechanics*, pp. 361 - 371.

Hoang, N. T., Rediniotis, O. K., and Telionis, D. P., 1993, "Three-D LDV Measurements Over a Delta Wing in Pitch-Up Motion," AIAA Paper No. 93-0185.

Hoang, N. T., Rediniotis, O. K., and Telionis, D. P., 1993, "3-D LDV Measurements over Delta Wings in Pitch-Up Motions," AIAA Paper 93-0255.

Hoang, N. T., Wetzel, T. G. and Simpson, R. L. , 1994, "Unsteady Measurements Over a 6:1 Prolate Spheroid Undergoing a Pitch-up Maneuver," AIAA Paper No. 94-0197.

Hoffler, K. D., Rao, D. M., and Frassinelli, M. C., 1985, "Basic Studies on Delta Wing Flow Modifications by Means of Apex Fences," *Vortex Flow Aerodynamics-Volume 1*, NASA Conference Publication 2416.

Hoffler, K. D. and Rao, D. M., 1985, "An Invetstigation of the Tabbed Vortex Flap," Journal of Aircraft, Vol. 22, No. 6, June 1985, pp. 490-497.

Holman, J. P., 1984, *Experimental Methods For Engineers*, McGraw-Hill Book Company, New York, New York

Hubner, J.P. and Komerath, N. M., 1994, "Visualization of Quasi-Periodic Structures in a Vortex Flow," AIAA Paper No. 94-0624.

Jackson, P. (Editor), 1995, *Jane's All the World's Aircraft 1995-1996*, Jane's Information Group, Surrey, UK

Kegelman, J. T. and Roos, F. W., 1990, "The Flowfields of Bursting Vortices Over Moderately Swept Delta Wings," AIAA Paper No. 90-0599.

Keller, J. J., Egli, W., and Althaus, R., 1988, "Vortex Breakdown as a Fundamental Element of Vortex Dynamics," *Fluid Dynamics Research*, No. 3, pp. 31-42.

Klute, S., Rediniotis, O. K., and Telionis, D. P., 1996, "Flow Control Over a Maneuvering Delta Wing at High Angles of Attack," *AIAA Journal*, Vol. 34, No. 4, pp. 662-668.

Koromilas, C. and Telionis, D. P., 1980, "Unsteady Laminar Separation, an Experimental Study," *Journal of Fluid Mechanics*, Vol. 94, pp. 347-384.

Lamar, J. E. and Campbell, J. F., 1983, "Recent Studies at NASA-Langley of Vortical Flows Interacting with Neighboring Surfaces," *Vortex Flow Aerodynamics*, AGARD Conference Proceedings 342, Paper No. 10.

Lamar, J. E., 1986, "Nonlinear Lift Control at High Speed and High Angles of Attack using Vortex Flow Control Technology." AGARD-R-740 Special Course on Fundamentals of Fighter Aircraft Design.

Lambourne, N. C. and Bryer, D. W., 1961, "The Bursting of Leading Edge Vortices --- Some Observations and Discussion of the Phenomenon," Reports and Memorandum No. 3282.

Lee, M. and Ho, Chih-Ming, 1990, "Lift Force on Delta Wings," *Applied Mechanics Review*, Vol. 43, pp. 209-221.

Leibovich, S., 1984, "Vortex Stability and Breakdown: Survey and Extension," *AIAA Journal,* Vol. 22, pp. 1192-1206.

Levich, E., and Tsinober, A., 1993, "On the Role of Helical Structures in Three Dimensional Turulent Flow," *Physical Letters,* Vol. 93A, No. 6, pp. 293-324.

Levy, Y., Degani, D., and Seginer, A., 1990, "Graphical Visualization of Vortical Flows By Means of Helicity," *AIAA Journal,* Vol. 28, No. 8, pp. 1347-1352.

Lin, J. C. and Rockwell, D., 1995, "Transient Structure of Vortex Breakdown on a Delta Wing," *AIAA Journal*, Vol. 33, No. 1, pp. 6-12.

Lowson, M. V., 1989, "Visualization Measurements of Vortex Flows," AIAA Paper No. 89-0191.

Lowson, M. V., Riley, A. J., and Swales, C., 1995, "Flow Structures over Delta Wings," AIAA Paper No. 95-0586.

Lugt, H. J., 1979, "The Dilemma of Defining a Vortex," *Recent Developments in Theoretical and Experimental Fluid Mechanics,* Springer-Verlag, New York, p. 309.

Magness, C., Robinson, O. and Rockwell, D., 1993, "Laser-Scanning Particle Image Velocimetry Applied to a Delta Wing in Transient Maneuver," *Experiments in Fluids*, Vol. 15, No. 3, pp. 159-167.

Magness, O., Robinson, O., and Rockwell, D., 1993, "Instantaneous Topology of the Unsteady Leading Edge Vortex at High Angle of Attack," *AIAA Journal,* Vol. 31, No. 8, pp. 1384-1391.

Marchman III, J. F., 1981a, "Aerodynamics of Inverted Leading-Edge Flaps on Delta Wings," *Journal of Aircraft*, Vol. 18, pp. 1051-1056.

Marchman III, J. F., 1981b, "Effectiveness of Leading-Edge Vortex Flaps on 60 and 75 Degree Delta Wings," *Journal of Aircraft*, Vol. 18, pp. 280-286.

Meyers, J. F. and Hepner, T. E., 1988,"Measurement of Leading Edge Vortices From a Delta Wing Using a Three Component Laser Velocimeter," AIAA Paper No. 88-2024.

Moffatt, H. K., 1969, "The Degree of Knottedness of Tangled Vortex Lines," *Journal of Fluid Mechanics,* Vol. 35, Part 1, pp. 117-129.

Moffet, K. H., and Tsinober, A., 1992, "Helicity in Laminar and Turbulent Flow, " *Annual Review of Fluid Mechanics,* 24:281-312.

Nelson, R. C. and Visser, K. D., 1983, "Breaking Down the Delta Wing Vortex: The Role Of Vorticity in the Breakdown Process," *Vortex Flow Aerodynamics*, AGARD Conference 494, Paper No. 21.

Payne, F. M. and Nelson, R. C., 1985, "An Experimental Investigation of Vortex Breakdown on a Delta Wing," *Vortex Flow Aerodynamics-Volume 1*, NASA Conference Publication 2416.

Payne, F. M., Ng, T. T. and Nelson, R. C., 1989, "Seven hole probe measurement of leading edge vortex flows," *Experiments in Fluids*, Vol. 7, pp. 1-8.

Payne, F. M., Ng, T. T. and Nelson, R. C., 1988, "Visualization and Wake Surveys of Vortical Flow over a Delta Wing," *AIAA Journal*, Vol. 26, pp. 137-143.

Payne, F. M., 1987, "The Structure of the Leading Edge Vortex Flows Including Vortex Breakdown," Ph.D. Dissertation, University of Notre Dame, IN.

Perry, A. E. and Chong, M. S., "A Description of Eddying Motions and Flow Patterns Using Critical-Point Concepts", *Annual Review of Fluid Mechanics*, 1987, 19:125-55.

Rao, D. M. and Campbell, J. F., 1987, "Vortical Flow Management Techniques," *Progress in Aerospace Sciences*, Vol. 24, No. 3, pp. 173-244.

Rao, D. M., 1979, "Leading Edge Vortex Flap Experiments on a Delta Flap Wing," NASA CR-159161.

Rao, D. M., 1980, "Leading Edge Vortex Flaps for Enhanced Subsonic Aerodynamics of Slender Wings," Proceedings of the 1980 International Council of Aeronautical Sciences, Munich, Germany.

Rao, D. M., 1983, "Vortical Flow Management for Improved Configuration Aerodynamics – Recent Experiences," *Vortex Flow Aerodynamics*, AGARD Conference 494, Paper No. 30.

Rao, D. M., 1985, "Towards an Advanced Vortex Flap System - the Cavity Flap," *Vortex Flow Aerodynamics-Volume 1*, NASA Conference Publication 2416.

Rao, D. M., 1990, "Potential of Segmented Vortex Flaps for Post-Stall Controllability," NASA High-Angle-of-Attack Technology Conference.

Reddy, C. S., 1981, "Effect of Leading-Edge Vortex Flaps on Aerodynamic Performance of Delta Wings," *Journal of Aircraft*, Vol. 18, pp. 796-798.

Rediniotis, O. K., Stapountzis, H., and Telionis, D. P., 1989, "Natural Periodic Phenomena over Delta Wings at High Angles of Attack," AIAA Paper No. 89-1923.

Rediniotis, O. K., Hoang, N. T., and Telionis, D. P., 1991, "Multi-sensor Investigations of Delta Wing High-Alpha Aerodynamics," AIAA Paper No. 91-0735.

Rediniotis, O. K., 1992, "The Transient Development of Vortices Over Delta Wings," Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Department of Engineering Science and Mechanics, Blacksburg, Virginia.

Rediniotis, O. K., Klute, S. M., Hoang, N. T., and Telionis, D. P., 1994, "Pitch-Up Motions of Delta Wings," *AIAA Journal*, Vol. 32, No. 4, pp. 716-725.

Rockwell, D., 1993, "Three- Dimensional Flow Structures on Delta Wings at High Angle-of-Attack: Experimental Concepts and Issues," AIAA Paper No. 93-0550.

Roos, F. D. and Kegelman, J. T., 1990, "An Experimental Investigation of Swept-Angle Influence on Delta-Wing Flows," AIAA Paper No. 90-0383.

Roos, W. F. and Kegelman, J. T., 1990, "Recent Explorations of Leading-Edge Flowfields," Presented at NASA-Langley High-Angle-of-Attack Technology Conference.

Sarpkaya, T., 1971, "On Stationary and Traveling Vortex Breakdowns," *Journal of Fluid Mechanics*, Vol. 45, p. 545.

Sarpkaya, T., 1979, "Vortex-Induced Oscillation," *Journal of Applied Mechanics*, Vol. 46, pp. 241-258.

Sarpkaya, T., 1995, "Vortex Breakdown and Turbulence," AIAA Paper No. 95-0433

Seider, G., 1984, "The Design, Construction, and Calibration of a Low Speed Wind Tunnel," Report filed for Dr. D. P. Telionis on behalf of the VPI-ESM Fluid Mechanics Laboratory.

Squire, L.C., 1961, "An Experimental Characteristics of Some Plane and Cambered 65 Delta Wings at Mach Numbers from 0.7 to 2.0," ARC Reports and Memoranda No. 3305.

Telionis, D. P., Mathioulakis, D. S., Kim, B. K. and Jones, G. S., "Calibration of the ESM Water Tunnel," VPI&SU Engineering Report No. VPI-E-86-23, 1986.

Taylor, J. R., 1982, *An Introduction to Error Analysis*, University Science Books, Oxford University Press, Mill Valley, California.

Thompson, S. A. , Batill, S. M., and Nelson, R. C., 1991, "Seperated Flowfield on a Slender Wing Undergoing Transient Pitching Motions", *Journal of Aircraft*, Vol. 28, No. 8, pp. 489-495.

Visbal, M. and Gordnier, R., 1994a, "Crossflow Topology of Vortical Flow," *AIAA Journal*, Vol. 32, No. 5, pp. 1085 - 1087.

Visbal, M. R. and Gordnier, R.F., 1994b, "Parametric Effects on Vortex Breakdown over a Pitching Delta Wing," AIAA Paper No. 94-0538.

Visbal, M. R., 1995, "Computational and Physical Aspects of Vortex Breakdown on Delta Wings," AIAA Paper No. 95-0585.

Visser, K. D. and Nelson, R. C., 1993, "Measurements of Circulation and Vorticity in the Leading-Edge Vortex of a Delta Wing," *AIAA Journal*, Vol. 31, No. 1, pp. 104 – 111.

Washburn, A. C., 1992, "Effects of External Influences on Subsonic Delta Wing Vortices," AIAA Paper No. 92-4033.

Washburn, A. E. and Visser, K. D., 1994, "Evolution of Vortical Structures in the Shear Layer of Delta Wings," AIAA Paper No. 94-2317.

Wedemeyer, E., 1982, "Vortex Breakdown," in *High-Angle-of-Attack Aerodynamics*, J. F. Wendt, ed., AGARD/VRI *Lecture Series*, No. 121.

Wentz, W. H. and Kohlman, D. L., 1971, "Vortex Breakdown on Slender Sharp-Edge Delta Wings," *Journal of Aircraft*, Vol. 8, No 3, March 1971.

Wetzel, T. G., 1996, "Unsteady Flow over a 6:1 Prolate Spheriod," Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Department of Aerospace and Ocean Engineering, Blacksburg, Virginia.

Wilder, M. C., 1992, "Airfoil-Vortex Interaction and the Wake of an Oscillating Airfoil," Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Department of Engineering Science and Mechanics, Blacksburg, Virginia.

APPENDIX A: MECHANICAL DRAWINGS FOR THE MODELS

This appendix contains images of the original mechanical drawing for the models used in this investigation. Most of the original mechanical drawings were done by hand and in pencil before being sent to various machine shops. Unless otherwise noted, all the mechanical drawings are NTS - “not to scale”.

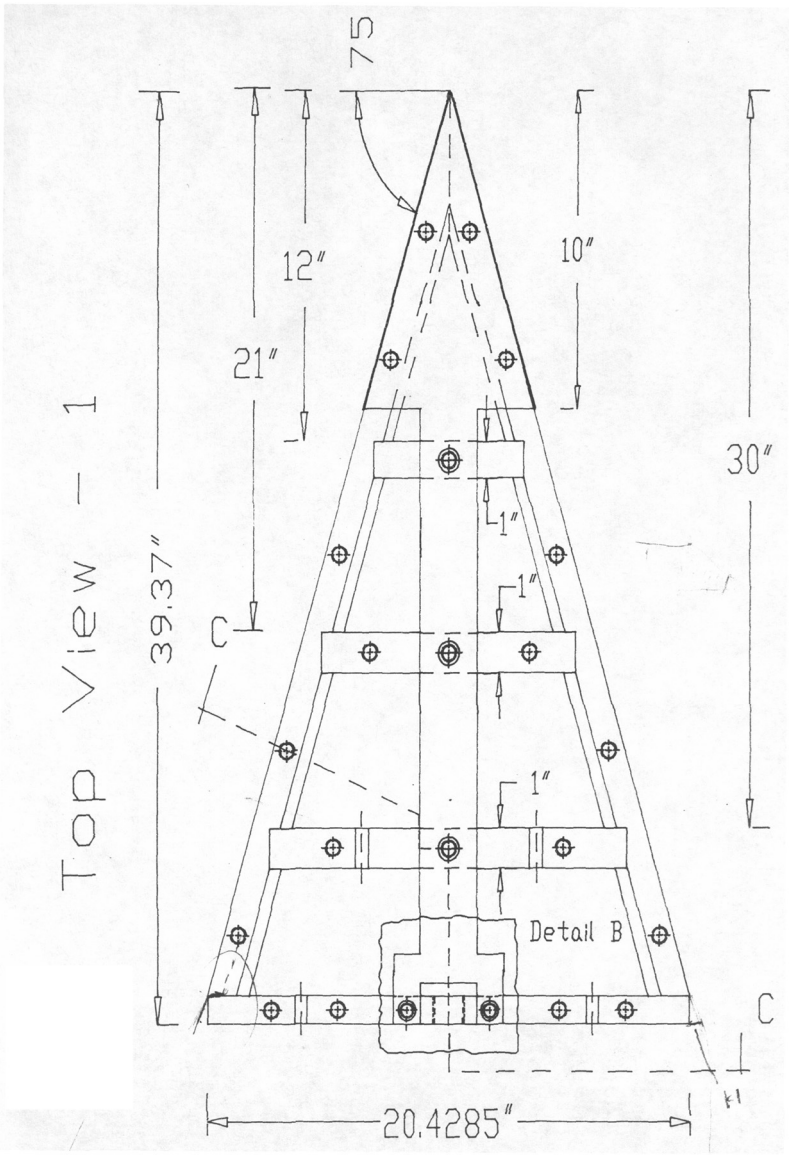


Figure A.1: The 1-meter model original plan, Top View 1 (Rediniotis, 1992).

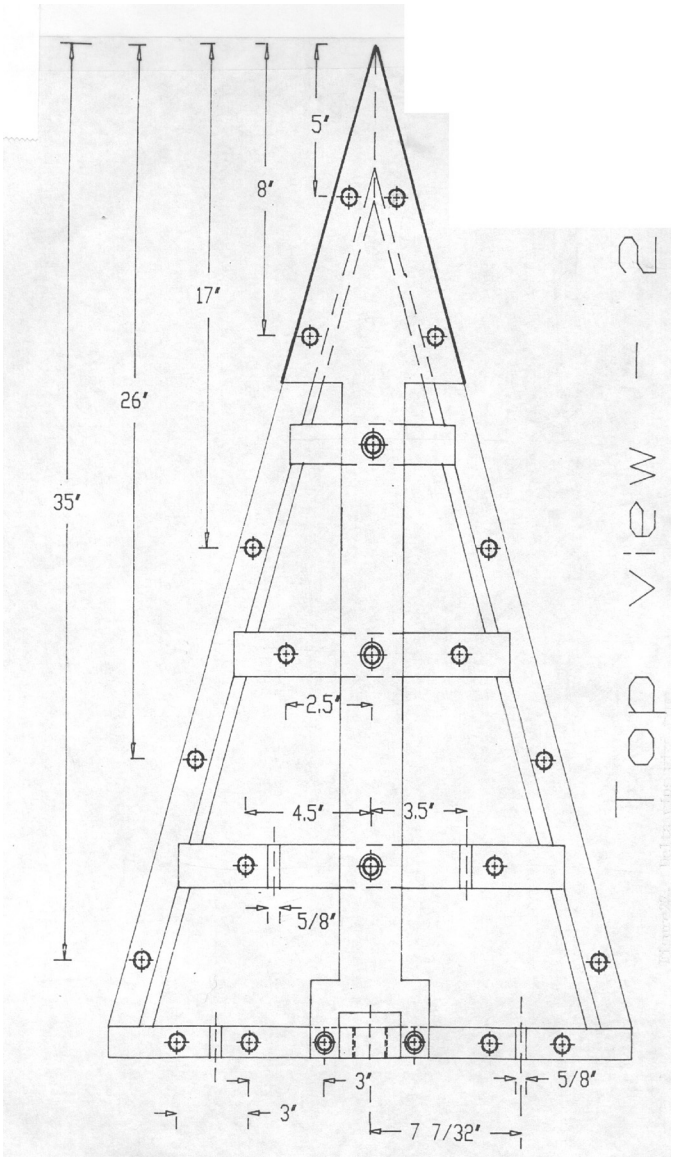


Figure A.2: The 1-meter model original plan, Top View 2 (Rediniotis, 1992).

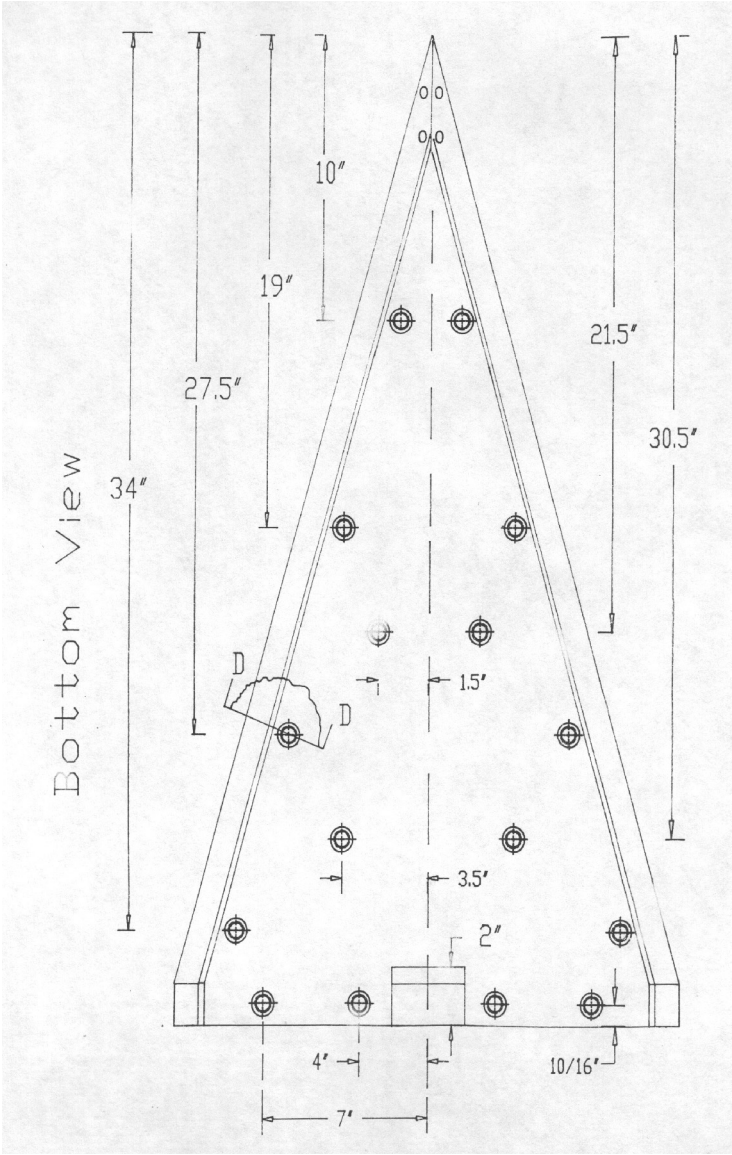


Figure A.3: The 1-meter model original plan, Bottom View 1 (Rediniotis, 1992).

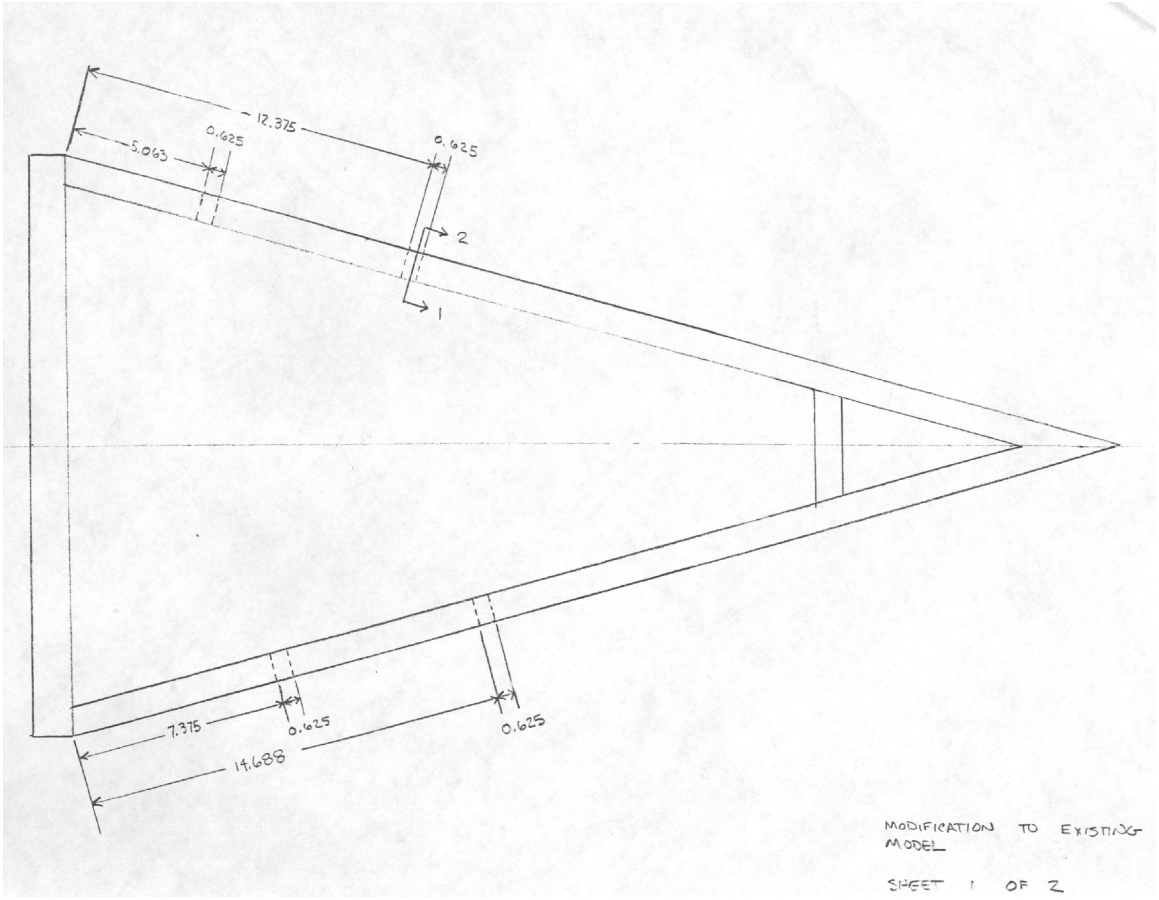


Figure A.4: Modifications to the 1-meter model original plan to accommodate the cavity flaps, Sheet 1.

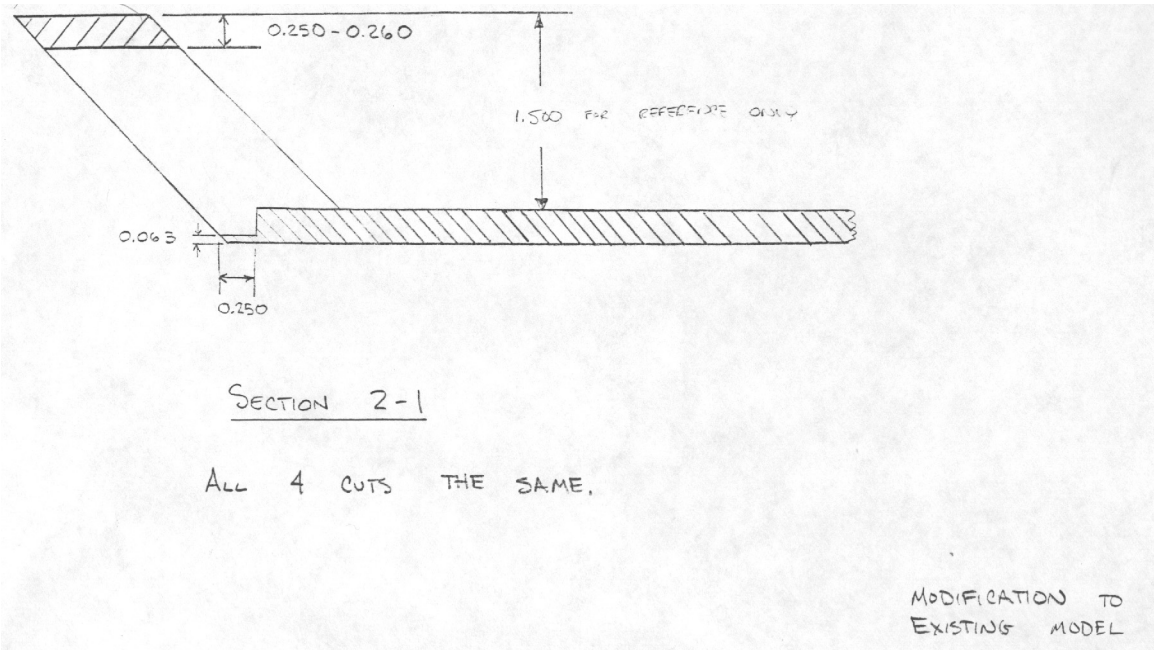


Figure A.5: Modifications to the 1-meter model original plan to accommodate the cavity flaps, Sheet 2.

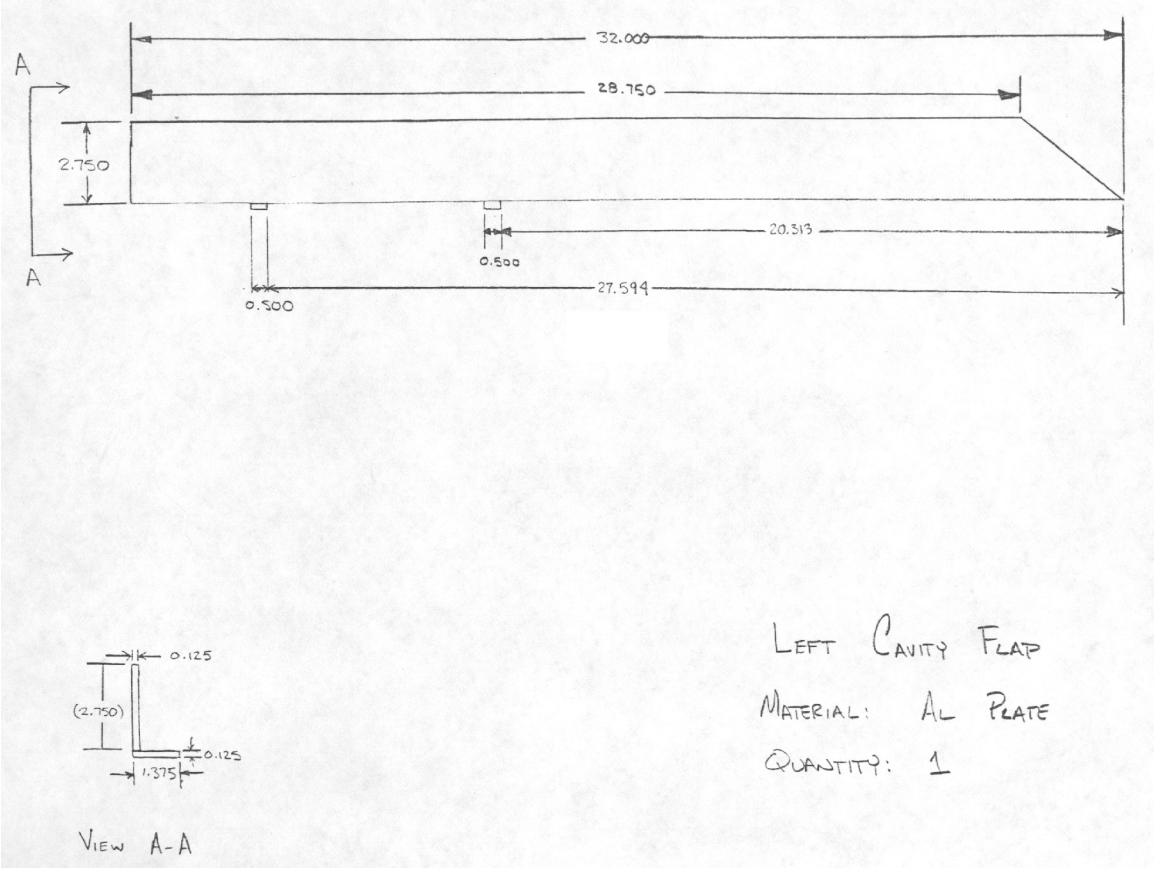


Figure A.6: Modifications to the 1-meter model original plan to accommodate the cavity flaps. "The left cavity flap" and its mounting bracket.

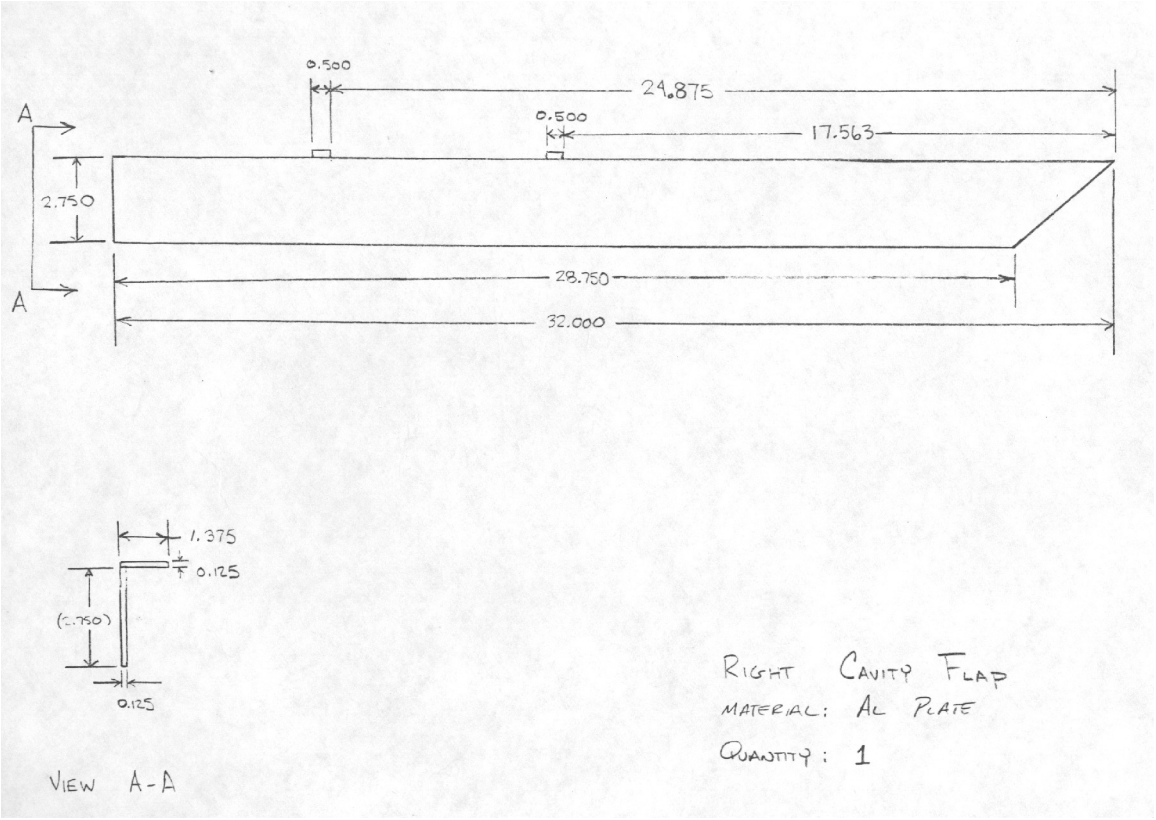


Figure A.7: Modifications to the 1-meter model original plan to accommodate the cavity flaps. "The right cavity flap" and its mounting bracket.

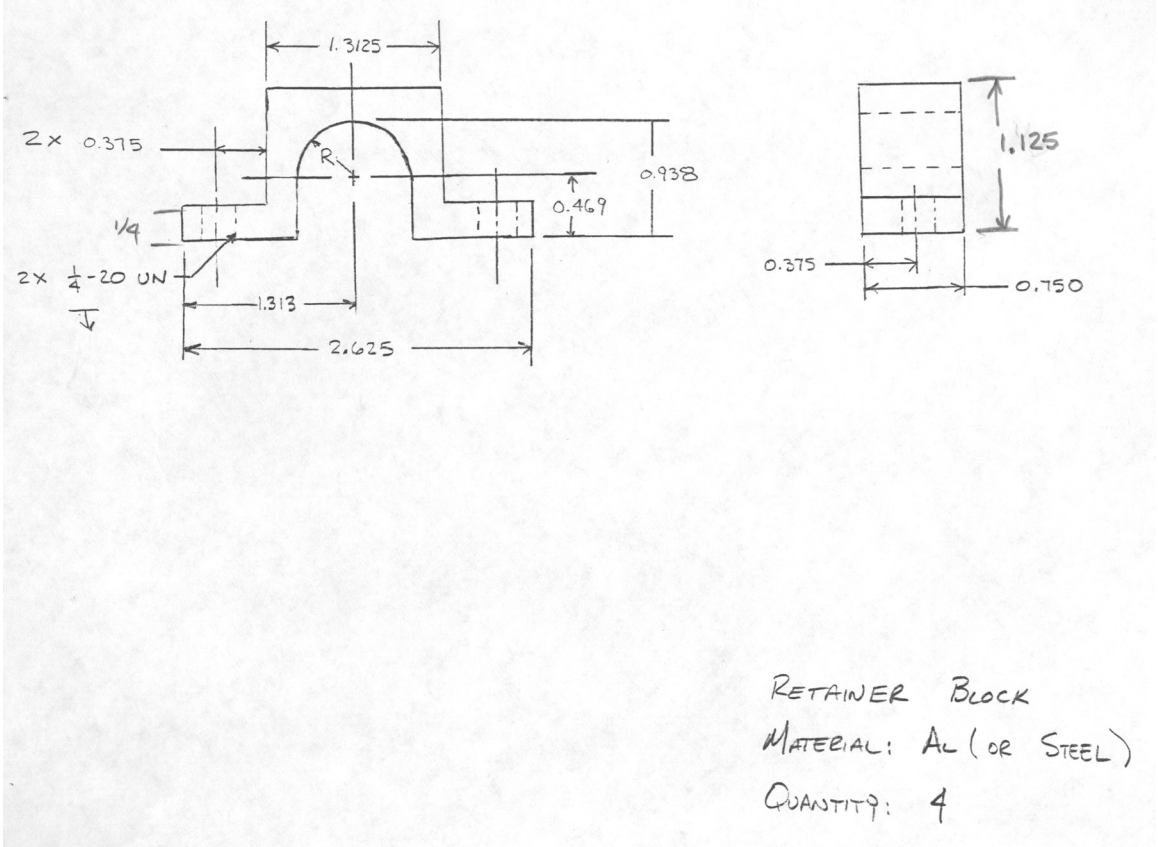


Figure A.8: Modifications to the 1-meter model original plan to accommodate the cavity flaps. Actuator Mounting Blocks. The mounting blocks for the Bimba air cylinders that deployed the flaps. These held the air cylinders securely to the floor of the model.

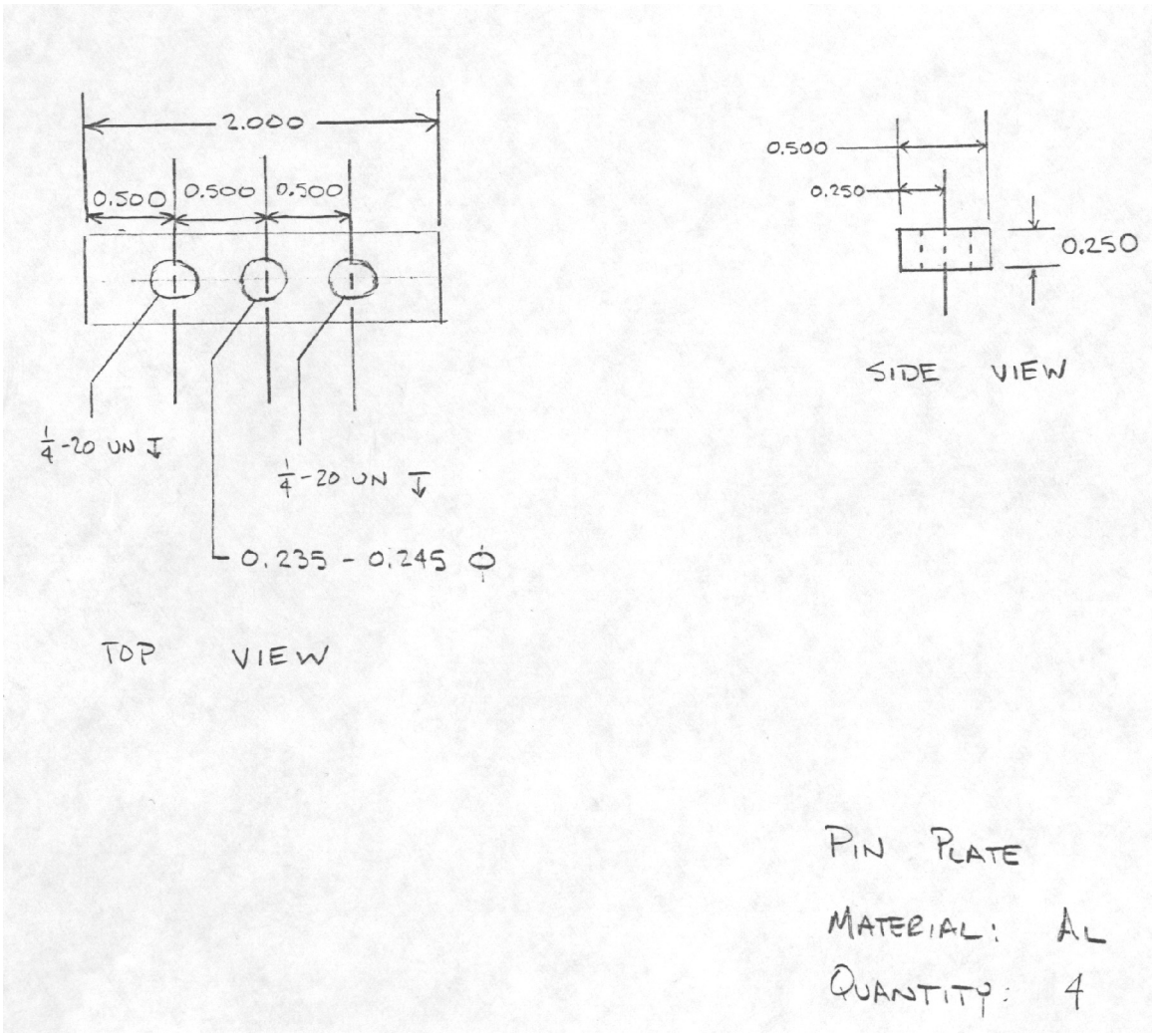


Figure A.9: Modifications to the 1-meter model original plan to accommodate the cavity flaps. Pin Blocks. The Pin Blocks held the stainless steel pins securely to the floor of the model. The air cylinders had holes through which the pins were inserted.

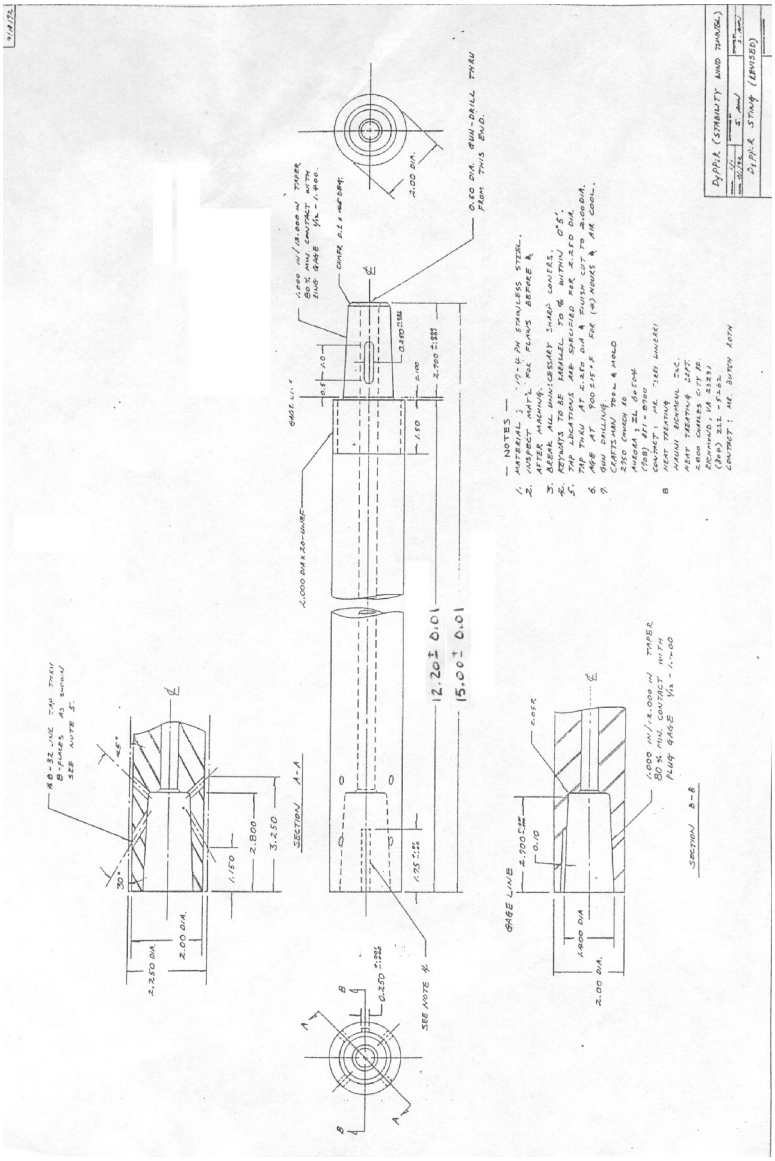


Figure A.10: Mechanical Drawing for the sting used in this investagation. Drawing is nothing more than a modified version of the "Official" DyPPiR sting design by Ahn (1992). The only difference is the length of this sting.

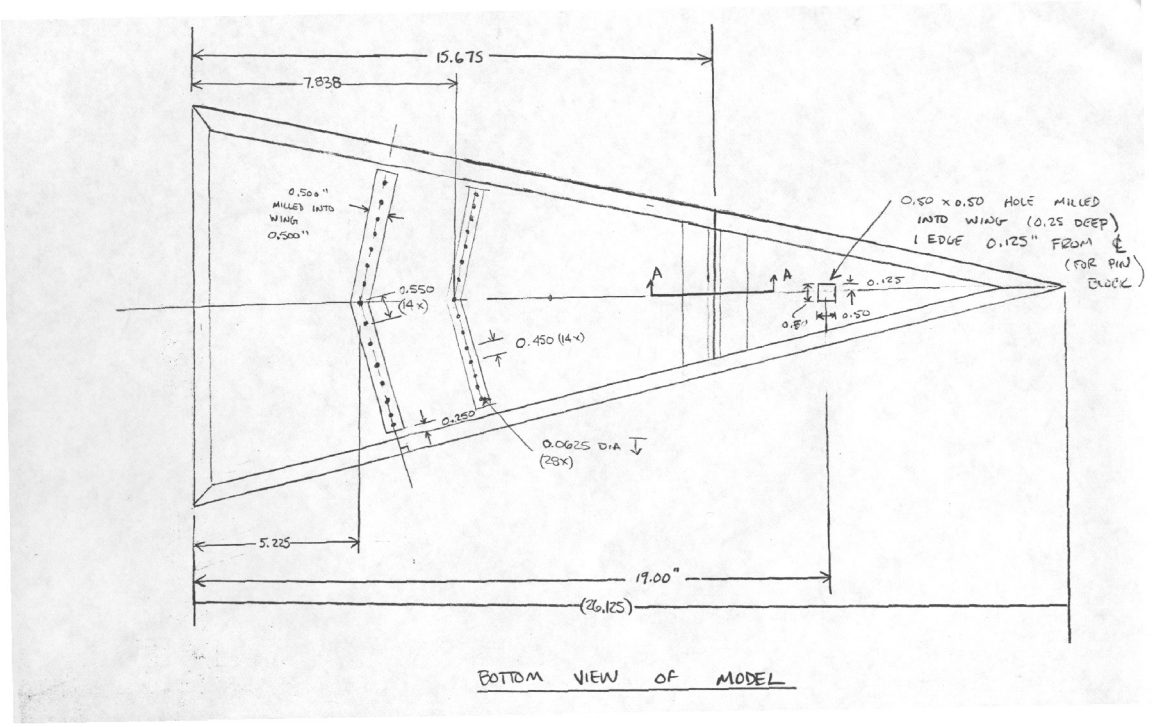


Figure A.11: Mechanical drawing for the modifications to the existing 0.66-meter chord wing to accommodate apex flaps (Sheet 1).

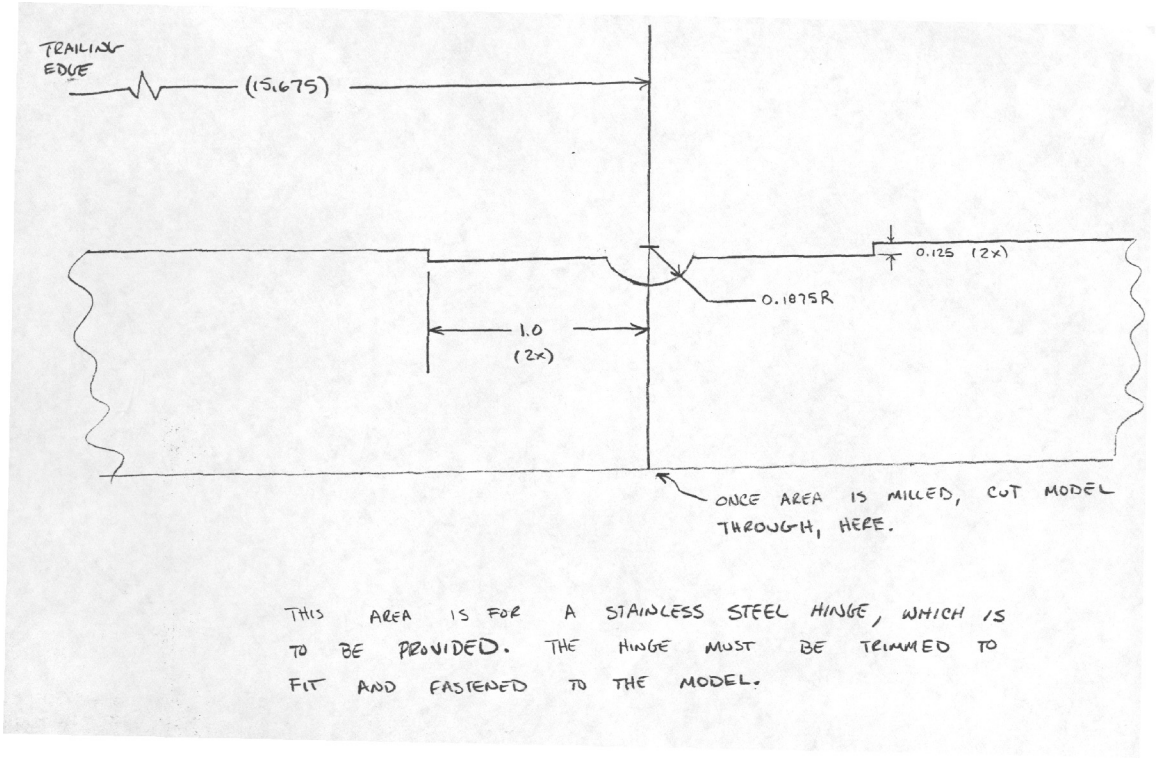


Figure A.12: Detail of the modifications required to accommodate the hinge for the apex flap.

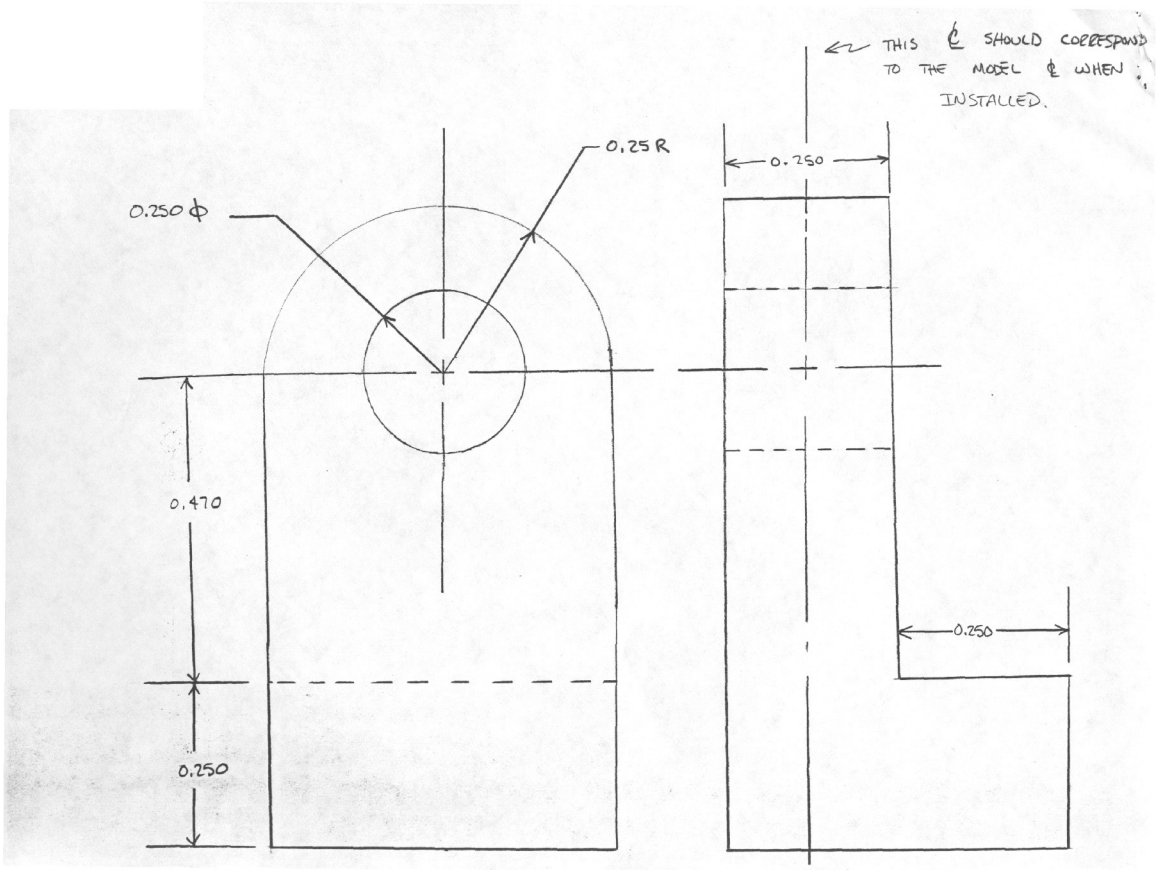


Figure A.13: Pin design for the apex flap modification to the 0.66-meter model. Pin is accepted into the clevis of the air cylinder.

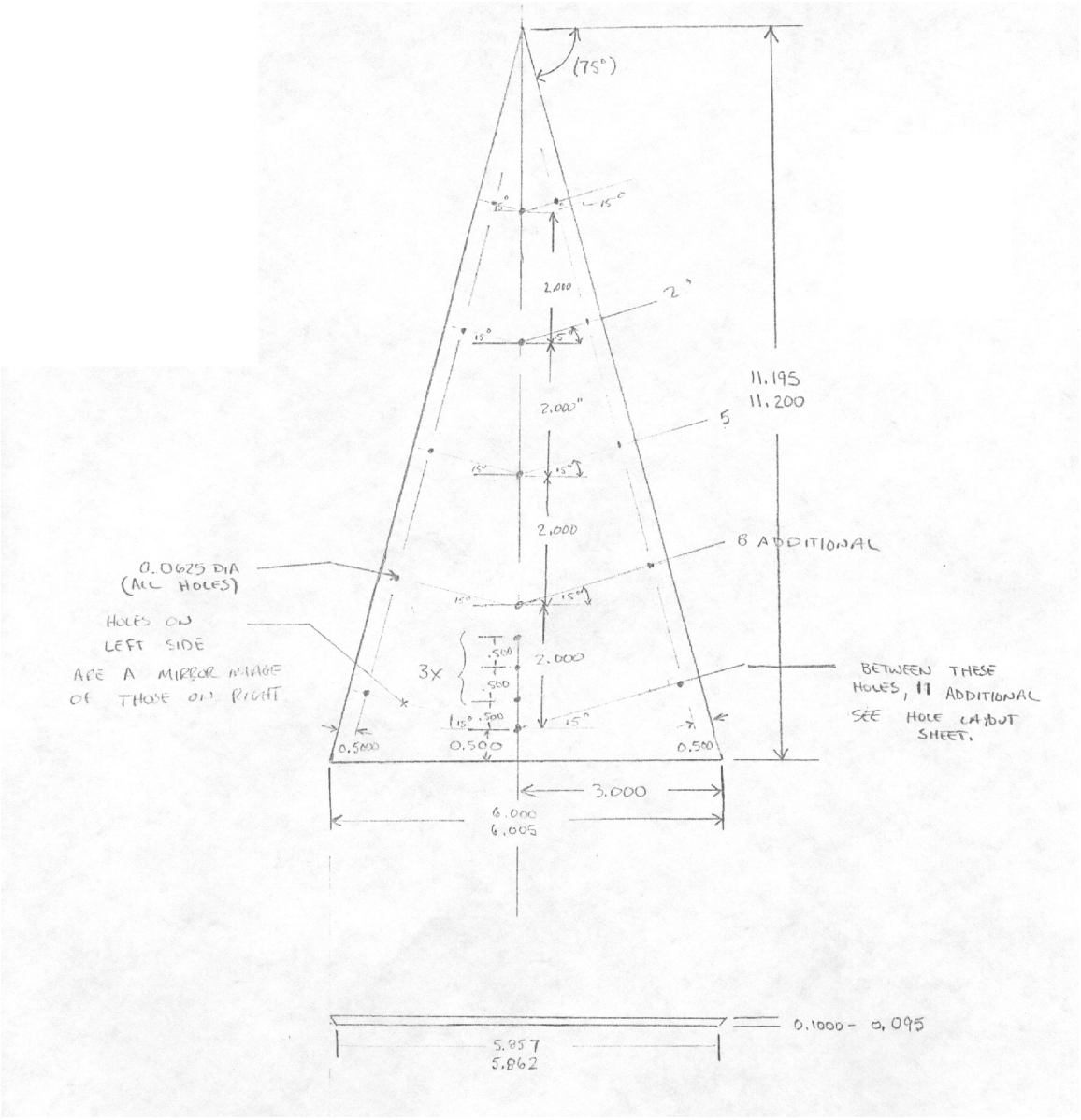


Figure A.14: Mechanical drawing for the 0.28-meter chord model. Sheet 1- Top View.

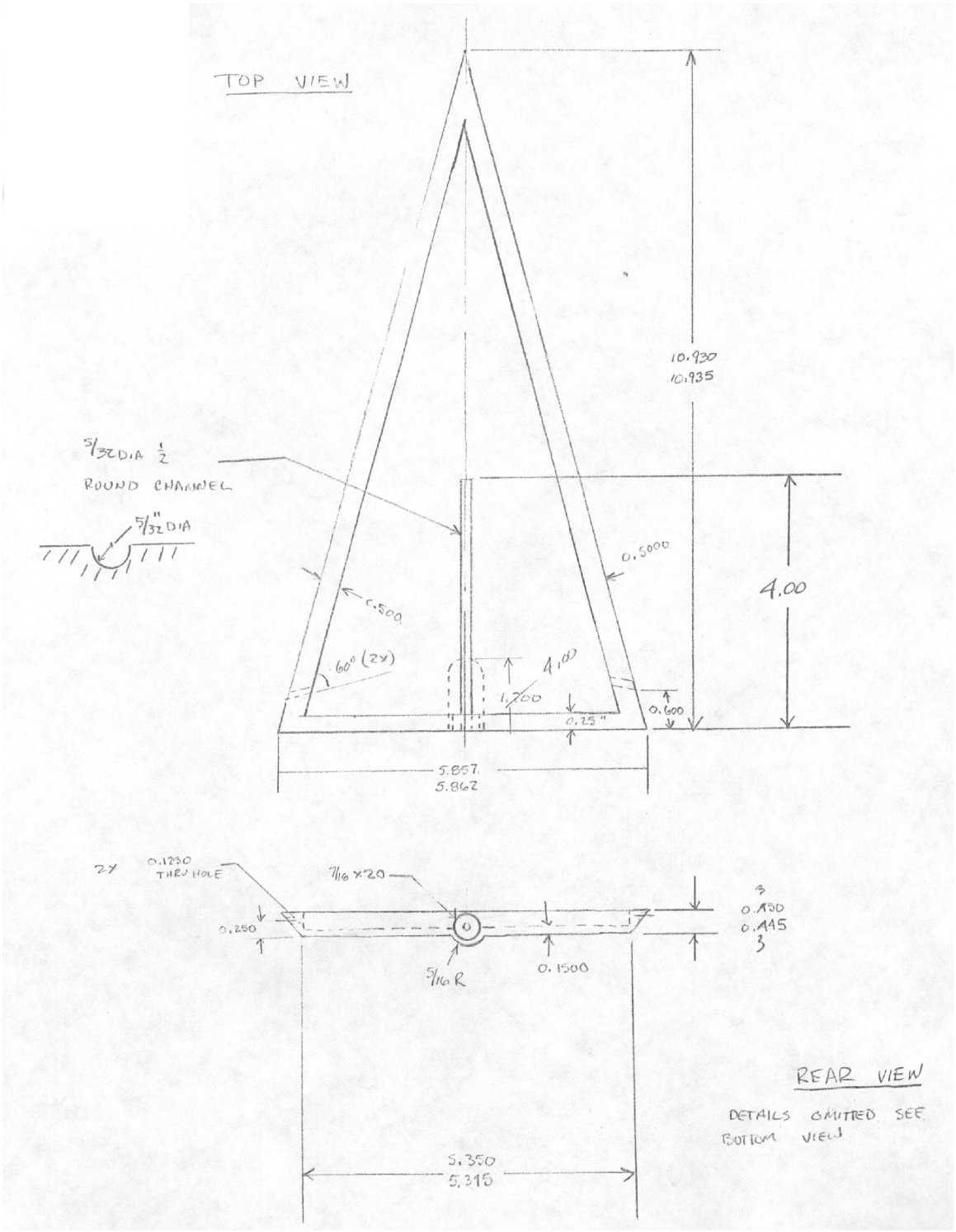


Figure A.15: Mechanical drawing for the 0.28-meter chord model. Sheet 2- Bottom View.

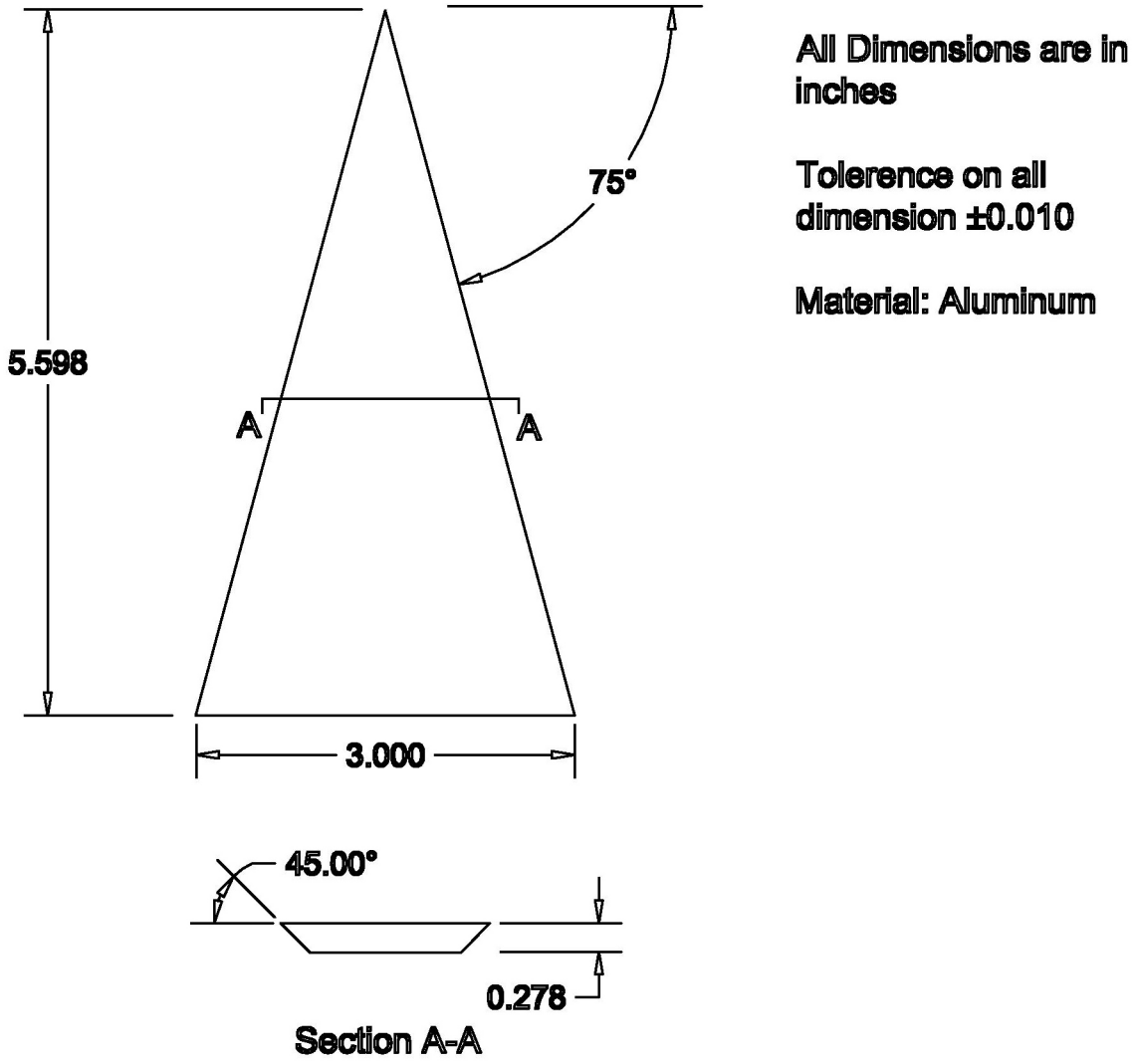


Figure A.16: Mechanical Drawing for the LDV Models.

APPENDIX B: *SWINGMAN* CODE LISTING

This appendix contains the code listing for the "Swingman" program which ran on the ISA PC while taking data at the ESM Wind Tunnel facility. Swingman communicates with the IBM PS/2 Model 60, which runs the motor controlling the pitch, acquires the LVDT signals, both to find the start position and during the motion, and acquires the ESP data. The code supports the ESPio-1 board and the RC Electronics ISC-16 board. The code contains provisions to calibrate the ESP and to acquire data from it upon receipt of a trigger. This trigger was provided by the ISA machine itself via one of the pins of the parallel port that is broken-out of the connector and sent to the "External Trigger" input of the RC Electronics board and indicated that the motion had begun. A listing of "screen.h" is included at the bottom of this listing.

```
//      SWINGMAN.C   The Pitch Up Swing Manager!
//         (c) 1994-5 Norman W. Schaeffler
//
//  Includes: Communication with Model 60 via Parallel Port
//            ESP Routines
//            RC Electronics Routines

// C includes
#include <graph.h>
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <math.h>

// Project Includes
#include "screen.h"
```

```
// Defines
#define NoCursor  0x2000
#define Underline 0x0707
#define CursorHome   _settextposition( 23, 2 )
#define StopCommand       0
#define SlowIdleCommand  10
#define FastIdleCommand  20
#define PitchCommand     65
#define ShutDownCommand 255
#define TriggerCommand  128

// Function Protoypes
         float  Get_LVDT_Value( void );
          void  AcquireRCDataOnTrigger( void );
          void  SaveRCData( void );
          void  FindStartPosition( void );
          void  Delay( long );
          void  WriteParallel( int );
          void  BuildScreen( void );
          void  GetParameters( void );
          void  GetDate( char date_string[40] );
          void  ShutDownModel60( void );
          void  SetUpESP( void );
          void  ESP_TestSuite( void );
         float  Address_and_Sample( void );
          void  TriggerRC_And_AcquirePressures( void );
          void  WritePressures( void );
          void  ReadStartPosition( void );
          void  We_Are_Triggered( void );
           int  Round( float );
          void  OutlineScreen( void );
          void  CalibrateESP( void );
          void  PerformESPCalibration( void );
          void  Compute_Coefficients( void );
          void  Display_Coefficients( void );
          void  SaveESPData( void );
          void  Freestream( void );
unsigned  char  HI( unsigned int );
unsigned  char  LOW( unsigned int );


// Structures
struct ESP_Info
   {
    char date[40];
    int  NumberOfEnsembles;
    int  NumberOfChannels;
    int  NumberOfPorts;
    int  SamplingRate;
    int  Number_Of_Samples;
    int  ESP_SamplingRate;
    int  Number_Of_ESP_Samples;
   };
```

```
// Global Variables
struct ESP_Info info;
            int NumberOfCalPoints, Number_Of_Samples, NumberOfChannels;
            int NumberOfEnsembles, BufferSize, SamplingRate;
            int FlagFor32, DurationOfSampling, BaseFlag;
           char filename[12], date[40], basename[10];
           char CommandString1[]="pkzip.exe";
           char CommandString2[60];
           FILE *DataFile;
            int ESP_PortNumber, ESP_OperationMode, NumberOfEnsembles;
            int NumberClockPulses, Number_Of_ESP_Samples, NumberOfPorts;
         float ESP_SamplingRate, ESP_BurstRate, StartVoltage;
        double Voltage[10][32], Pressure[10], Calibration_Coeffs[3][32];

            int RC_Data[200];     // Must Match Number_Of_Samples
            int hi[200][32],lo[200][32];
                                  // 1st is ESP_SamplingRate * ESP_Duration

void main( void )
{
   char buffer[12], ch;
   int i, j;

   _clearscreen( _GCLEARSCREEN );

// RC and ESP Stuff

   FlagFor32          = 1;      // 1 if 32 channel, 0 if 16 channel
   DurationOfSampling = 1.0;    // In Seconds
   SamplingRate       = 200;    // in Hz
   NumberOfChannels   = 1;
   Number_Of_Samples  = NumberOfChannels * SamplingRate * DurationOfSampling;

// Do all the background information

   GetParameters();
   _settextcursor( NoCursor );
   ReadStartPosition( );
   ESP_TestSuite();
   CalibrateESP();
   Freestream();
   SetUpESP();


   if ( (DataFile = fopen(filename,"wb")) == NULL )
     {
     _clearscreen( _GCLEARSCREEN );
     printf("Error opening file %s\nProgram Aborted\n", filename);
     outp( 0x0378, StopCommand );
     ShutDownModel60();
     exit(0);
     }
```

```
// Fill and Save info structure and Calibration Coeffs.

   GetDate( info.date );
   info.NumberOfEnsembles     = NumberOfEnsembles;
   info.NumberOfChannels      = NumberOfChannels;
   info.NumberOfPorts         = NumberOfPorts;
   info.SamplingRate          = SamplingRate;
   info.Number_Of_Samples     = Number_Of_Samples;
   info.ESP_SamplingRate      = ESP_SamplingRate;
   info.Number_Of_ESP_Samples = Number_Of_ESP_Samples;

   fwrite( &info, sizeof( info ), 1, DataFile );
   fwrite( Calibration_Coeffs, sizeof( Calibration_Coeffs ), 1, DataFile);

   BuildScreen();
   _settextposition( 12, 59);
   _outtext("Idle.              ");  // Motor Status

   for( i = 0; i < NumberOfEnsembles;  i++)
     {
     _settextposition( 9, 39 );
     sprintf(buffer,"%4d", i+1);
     _outtext( buffer );

     FindStartPosition();
     _settextposition( 12, 59);
     _outtext("Idle.              ");  // Motor Status
     Delay( 5000L );

     AcquireRCDataOnTrigger();
     TriggerRC_And_AcquirePressures();
     outp( 0x0378, StopCommand );

     Delay( 5000L );
     SaveRCData();
     SaveESPData();
     }

   FindStartPosition();
   ShutDownModel60();
   fclose( DataFile );

// Try to created the base of the filename

   strcpy( basename, "");

   for ( i = 0; i < 8; i++)
    {
     ch = filename[i];

     if ( ch == '.' )
        break;
```

```
     else
        basename[i] = ch;
     }

   basename[i] = 0x0;

   _settextcursor( Underline );
   _clearscreen( _GCLEARSCREEN );

   sprintf(CommandString2, "a:\\%s.zip %s\n", basename, filename );
   printf("\n\nCommand String 1: %s\n", CommandString1 );
   printf("\n\nCommand String 2: %s\n", CommandString2 );

}

void GetParameters( void )
{
   char ch;

   _settextcolor(15);
   _setbkcolor(0);
   _clearscreen( _GCLEARSCREEN );

   printf("\n\n\tEnter the number of ensembles -->\t");
   scanf("%d", &NumberOfEnsembles);

   NumberOfPorts = 16;

   printf("\tEnter the filename for the data  -->\t");
   scanf("%s", &filename);

   printf("\n\n\tStart Motor Manager on the Model 60 at this time.\n\n");
   printf("\tHit any key to continue ... ");

   while ( !kbhit() ) { }
   ch = getch();
   if ( ch == 0 ) ch = getch();

   outp( 0x0378, StopCommand );

}

void SetUpESP( void )
{
//
// ESP Operation Modes
// 160 - Multiple Readings, internal oscillator, software trigger
//  32 - Multiple Readings, internal oscillator, external trigger
//  64 - Multiple Readings, external oscillator
//
// ESP_SamplingRate is the frequency of the between bursts
// ESP_BurstRate is the frequency at which the ESP switches from port to
// port within a burst.
```

```
//
//
   float Exponent_Real, clock_freq, sample_freq;
   int   Exponent;
   float NumberClockPulsesAsFloat, Time_Of_ESP_Sampling;
   char ch;

   _clearscreen( _GCLEARSCREEN );

   ESP_OperationMode    =      32; //            !!!DO NOT FORGET!!!
   ESP_SamplingRate     =   200.0; // Hz.               TO
   Time_Of_ESP_Sampling =   1.000; // Seconds      Change hi and lo
   ESP_BurstRate        = 15625.0; // Hz.         array dimensions too!

   Exponent_Real = log( 500000.0 / ESP_BurstRate )/ log ( 2.0 );
   Exponent      = Round(Exponent_Real);

   clock_freq  = 1000000.0 / pow( 2.0, Exponent );
   sample_freq = clock_freq / 2.0;
   ESP_OperationMode += Exponent;

   NumberClockPulsesAsFloat = (1.0/ESP_SamplingRate-(NumberOfPorts-
1)/sample_freq)
                              * clock_freq;

   NumberClockPulses = Round( NumberClockPulsesAsFloat );


   ESP_SamplingRate = 1.0/(NumberClockPulses/2.0/sample_freq
                                          + (NumberOfPorts-1)/sample_freq);
   Number_Of_ESP_Samples = (int)( ESP_SamplingRate * Time_Of_ESP_Sampling );

}

void AcquireRCDataOnTrigger( void )
{
   int i, NumberOfMicroSec;
   float voltage, value, DeltaTime;

   DeltaTime = 1.0 / SamplingRate;
   NumberOfMicroSec = (int)( DeltaTime * 1000000.0 );

   outp(0x030B,0x0);     // Disabling Data Acqusition System

   for ( i=0; i<16; i++)  // Loading Multiplexer Table
      outp(0x0310,0x0);   // Channel 0 is being used

   outp(0x031A,0x08);   // Trigger channel 1, IRQ3 on post trigger delay
   outp(0x0311,0x00);   // Select Internal Clock
   outp(0x0308,0x0B);   // Trigger slope, external trigger, positive polarity

   outp(0x0307,0x74);     // set clock
   outp(0x0305, LOW(NumberOfMicroSec) );     // Load LSB
```

```
   outp(0x0305,  HI(NumberOfMicroSec) );     // Load MSB

   outp(0x0307,0x32);      // Set Number of Channels for each burst
   outp(0x0304,LOW(NumberOfChannels) );     // Load LSB
   outp(0x0304, HI(NumberOfChannels) );     // Load MSB

   outp(0x0307,0xB2);     // set post trigger delay for number of samples
   outp(0x0306, LOW(Number_Of_Samples) );     // Load LSB
   outp(0x0306,  HI(Number_Of_Samples) );     // Load MSB

   outp(0x0315,0x00);      // Select bank B
   outp(0x031B,0x03);      // reset bank A memory pointer for 2048 samples
   BufferSize = 2048;

   outp(0x030A,0x0);      // Enable Data Acquisition
   outp(0x030C,0x0);      // Enable Trigger Logic

   _settextposition( 12, 21);  // RC Status location
   _outtext("Waiting For Trigger ...");

}

void TriggerRC_And_AcquirePressures( void )
{
   int OldRead, NewRead, Samples, j, Sample, Port;

   outp( 0x3E6, ESP_OperationMode );     // Set mode and frequency

   outp( 0x3E5, 1);  // Initialize A/D subsystem to first port

   OldRead = 0;
   Samples = 0;

   We_Are_Triggered();

// Controls the triggering of the ESP and the RC
//
   outp( 0x0378, PitchCommand );       // ISA Computer LPT1
   Delay( 400L );   // This delay and the delay in Motorman determine
                    // how long the motor receives 10 Volts as a command
   outp( 0x0378, TriggerCommand );       // This is the trigger
// ************

   for( Sample = 0; Sample < Number_Of_ESP_Samples; Sample++ )
      {
      for( Port = 0; Port < NumberOfPorts; Port++)
         {
         outp( 0x3E5, Port );  // Address ESP
         Samples = 0;
         while( Samples < 2 )
           {
           NewRead = inp( 0x03E7 );
           NewRead = NewRead & 0x01;
```

```
            if ( (OldRead == 1) && (NewRead == 0 ))
               {
               hi[Sample][Port] = inp( 0x03E3 );
               lo[Sample][Port] = inp( 0x03E2 );
               Samples = Samples + 1;
               }

            OldRead = NewRead;
            }
         }

      for( j = 0; j < NumberClockPulses - 2; j++) // Dry sample till begining
         {                                        // of next sample run.
         Samples = 0;
         while( Samples < 1 )
            {
            NewRead = inp( 0x03E7 );
            NewRead = NewRead & 0x01;

            if ( (OldRead == 1) && (NewRead == 0 ))
               Samples = Samples + 1;

            OldRead = NewRead;
            }
         }
      }
}


void SaveRCData( void )
{
   int DataStart, data, i, hi,lo;
   float voltage, delta_time;

   _settextposition( 12, 21);  // RC Status location
   _outtext("Saving Data ...        ");

   DataStart  =  BufferSize - Number_Of_Samples - 1;
   delta_time = 1.0 / SamplingRate;

   outp(0x0314,0x03);     // select bank A

   for ( i=0; i< BufferSize; i++)
      {
      lo      = inp(0x031C);
      hi      = inp(0x031C);
      data    = lo + 256 * hi;
      if( i >= DataStart )
        RC_Data[i-DataStart] = data;
      }

   fwrite( RC_Data, sizeof( RC_Data ), 1, DataFile );
```

```
}

void SaveESPData( void )
{
   fwrite( hi, sizeof( hi ), 1, DataFile );
   fwrite( lo, sizeof( lo ), 1, DataFile );
}

float Address_and_Sample( void )
{
   float ESP_Voltage;
   int OldRead, NewRead, Samples, high, low, i;

   outp( 0x3E6, 165 );     // Set up for Internal Oscillator,
                           // Software Trigger, and Sampling Rate of
                           // about 15 kHz ( Othon uses 170 --> 1kHz )

   outp( 0x3E5, ESP_PortNumber - 1);  // address ESP and initialize A/D

   Delay( 50L );
   ESP_Voltage = 0.0000;
   for ( i = 0; i < 200; i++)
      {
      OldRead = 0;
      Samples = 0;

      while( Samples < 1 )
         {
         NewRead = inp( 0x03E7 );
         NewRead = NewRead & 0x01;

         if ( (OldRead == 1) && (NewRead == 0 ))
            {
            high = inp( 0x03E3 );
            low  = inp( 0x03E2 );
            Samples = Samples + 1;
            ESP_Voltage += ( 16 * high + low/16) * 10.0 / 4096 - 5.0;
            }

         OldRead = NewRead;
         }
      }

   ESP_Voltage = ESP_Voltage/200;
   ESP_Voltage = ESP_Voltage + 0.012 + 0.00346478 * ( 4.064 - ESP_Voltage );
   return( ESP_Voltage );
}

void ESP_TestSuite( void )
{
   int ESC_Flag = 0, MaxPortNumber;
   float PortVoltage;
   char ch, buffer[12];
```

```
_setbkcolor( 1 );
_settextcolor ( BrightWhite );

_clearscreen( _GCLEARSCREEN );
OutlineScreen();

_settextposition( 2, 24);
_outtext("E S P      T E S T     S U I T E");
_settextposition( 4, 15);
_outtext("Port");
_settextposition( 4, 47);
_outtext("Port");
_settextposition( 4, 25);
_outtext("Voltage");
_settextposition( 4, 57);
_outtext("Voltage");
_settextcolor ( Green + Blinking );
_settextposition( 22, 29 );
_outtext("Press ESC to Continue");
_settextcolor ( BrightWhite );

ESP_PortNumber = 1;
while( !ESC_Flag )
   {
   PortVoltage = Address_and_Sample();
   if( ESP_PortNumber/2*2 == ESP_PortNumber )   // Then it be even
      {
      _settextposition( ESP_PortNumber/2 + 4, 48 );
      sprintf( buffer, "%2d  ---> %6.3f", ESP_PortNumber, PortVoltage );
      _outtext( buffer );
      }
   else
      {
      _settextposition( ESP_PortNumber/2 + 5, 16 );
      sprintf( buffer, "%2d  ---> %6.3f", ESP_PortNumber, PortVoltage );
      _outtext( buffer );
      }
   ESP_PortNumber++;

   if( FlagFor32 > 0 )  // If true this is a 32 channel ESP
      MaxPortNumber = 32;
   else              //  If false this is a 16 Channel ESP
      MaxPortNumber = 16;

   if( ESP_PortNumber > MaxPortNumber ) ESP_PortNumber = 1;

   if( kbhit() )
      {
      ch = getch();
      if ( ch == 27 ) ESC_Flag = 1;
      if ( ch == 0 )  ch = getch();
      }
```

```
      }

   _clearscreen( _GCLEARSCREEN );
}

void CalibrateESP( void )
{
  int Port, i, j, Calibrate = 0, Response = 0;
  char buffer[50], tmpname[14], ch;
  float input;
  struct ESP_Info tmp;
  FILE *coeffs;

  _setbkcolor( 1 );
  _settextcolor ( BrightWhite );
  _clearscreen( _GCLEARSCREEN );

  OutlineScreen();

  _settextposition( 10, 10 );
  _outtext("Do you wish to calibrate the ESP or read in the coefficients");
  _settextposition( 11, 10 );
  _outtext("from an existing file?");

  while( !Response )
     {
     if ( Calibrate )       // Calibrate is selected option
        {
        _settextposition( 15, 30);
        _setbkcolor( 3 );
        _settextcolor( BrightWhite );
        _outtext("Calibrate");
        _settextposition( 15, 46);
        _setbkcolor( 1 );
        _settextcolor( Red );
        _outtext("Read");
        }
     else
       {
        _settextposition( 15, 30);
        _setbkcolor( 1 );
        _settextcolor( Red );
        _outtext("Calibrate");
        _settextposition( 15, 46);
        _setbkcolor( 3 );
        _settextcolor( BrightWhite );
        _outtext("Read");
        }

     while( !kbhit ) {}
     ch = getch();
     if ( ch == 13 ) Response = 1;           // ENTER
     if ( ch == 32 ) Calibrate = !Calibrate; // SPACE-BAR
```

```
     if ( ch ==  9 ) Calibrate = !Calibrate; // TAB
     if ( ch ==  0 ) ch = getch();
     }

   if ( Calibrate )
      PerformESPCalibration();
   else
     {
     _setbkcolor( 0 );
     _settextcolor ( BrightWhite );
     _clearscreen( _GCLEARSCREEN );
     printf("\n\n\tEnter the filename for the data  -->\t");
     scanf("%s", &tmpname);

     if ( (coeffs = fopen(tmpname,"rb")) == NULL )
        {
        _clearscreen( _GCLEARSCREEN );
        printf("Error opening file %s\nProgram Aborted\n", tmpname);
        outp( 0x0378, StopCommand );
        ShutDownModel60();
        exit(0);
        }

     fread( &tmp, sizeof( tmp ), 1, coeffs );
     fread( Calibration_Coeffs, sizeof( Calibration_Coeffs ), 1, coeffs);
     fclose( coeffs );
     Display_Coefficients();
     }

}

void PerformESPCalibration( void )
{
  int Port, i, j;
  char buffer[50], ch;
  float input;

  _settextcolor ( Green );
  _clearscreen( _GCLEARSCREEN );
  _settextposition( 12, 20 );
  _outtext("Apply 80 psi to C1 to set ESP in Calibration Mode");
  _settextposition( 14, 23 );
  _outtext("Open the REF line to the atmosphere");


  while ( !kbhit() )
  {}
  ch = getch();
  if ( ch == 0 )  ch = getch();

  _setbkcolor( 1 );
  _settextcolor ( BrightWhite );
  _clearscreen( _GCLEARSCREEN );
```

```
  OutlineScreen();

  _settextposition( 5, 10 );
  _outtext("Enter number of calibration points --> ");
  scanf("%d", &NumberOfCalPoints);
  _clearscreen( _GCLEARSCREEN );

  OutlineScreen();
  _settextposition( 7, 25 );
  sprintf(buffer,"Calibration Point:      of %d", NumberOfCalPoints);
  _outtext( buffer );

  for( i = 0; i < NumberOfCalPoints; i++)
     {
     _settextposition( 7, 46 );
     sprintf(buffer,"%2d", i+1);
     _outtext( buffer );

     _settextposition( 10 + i, 10 );
     _outtext("Enter Pressure reading from BaroCell --> ");
     scanf("%f", &input);
     Pressure[i] = input;

     for ( ESP_PortNumber = 1; ESP_PortNumber <= NumberOfPorts;
                                                  ESP_PortNumber++ )
        Voltage[i][ESP_PortNumber-1] = Address_and_Sample();

     }

  Compute_Coefficients();
  Display_Coefficients();

  _settextcolor ( Green );
  _clearscreen( _GCLEARSCREEN );
  _settextposition( 12, 20 );
  _outtext("Apply 80 psi to C2 to set ESP in Run Mode");
  _settextposition( 14, 23 );
  _outtext("Connect the REF line to the Tunnel static");


  while ( !kbhit() )
  {}
  ch = getch();
  if ( ch == 0 )  ch = getch();

  _settextcolor ( BrightWhite );


}

void Compute_Coefficients( void )
{
```

```
  int i, j, pv, row, col, Port;
  double a[3][3], r[3], r0, r1, r2;
  double sum_x, sum_x_squared, sum_x_cubed, sum_x_fourth;
  double sum_y, sum_xy, sum_yxx, d, pivot, factor;

  for ( Port = 0; Port < NumberOfPorts; Port++)
     {
     for ( i = 0; i < 3; i++)
        {
        for( j = 0; j < 3; j++ )
           a[i][j] = 0.0000000;

        r[i] = 0.0000000;
        }

     sum_x         = 0.00000;
     sum_x_squared = 0.00000;
     sum_x_cubed   = 0.00000;
     sum_x_fourth  = 0.00000;
     sum_y         = 0.00000;
     sum_xy        = 0.00000;
     sum_yxx       = 0.00000;

     for ( i = 0; i < NumberOfCalPoints; i++ )
        {
        sum_x         += Voltage[i][Port];
        sum_x_squared += Voltage[i][Port] * Voltage[i][Port];
        sum_x_cubed   += Voltage[i][Port] * Voltage[i][Port] *
Voltage[i][Port];
        sum_x_fourth  += Voltage[i][Port] * Voltage[i][Port]
                                          * Voltage[i][Port] *
Voltage[i][Port];

        sum_y         += Pressure[i];
        sum_xy        += Pressure[i] * Voltage[i][Port];
        sum_yxx       += Pressure[i] * Voltage[i][Port] * Voltage[i][Port];
        }

     a[0][0] = (double)(NumberOfCalPoints);
     a[0][1] = sum_x;
     a[0][2] = sum_x_squared;
     a[1][0] = sum_x;
     a[1][1] = sum_x_squared;
     a[1][2] = sum_x_cubed;
     a[2][0] = sum_x_squared;
     a[2][1] = sum_x_cubed;
     a[2][2] = sum_x_fourth;

     r[0]    = sum_y;
     r[1]    = sum_xy;
     r[2]    = sum_yxx;

//
```

```
// Solve the system of equations using Gauss-Jordan Method. Based on code
// from "FORTRAN 77 and Numerical Methods for Engineers" by G.J. Borse on
// Pgs. 416-417. Destroys both the a matrix and the r matrix.
//

     d = 1.0;

     for ( pv = 0; pv < 3; pv++)
        {
        if( fabs( a[pv][pv] ) < 1.0e-06 )
           {
           _clearscreen( _GCLEARSCREEN );
           printf("\n\n\n\t\tError! Zero Pivot detected during Matrix
Solve.\n");
           printf("\t\tESP probably not in Calibration Mode.\n");
           printf("\t\tProgram Terminating.\n\n");
           exit(0);
           }

        pivot = a[pv][pv];
        d     = d * pivot;

        for ( j = 0; j < 3; j++)
           a[pv][j] = a[pv][j] / pivot;

        r[pv]    = r[pv] / pivot;

        for( row = 0; row < 3; row++ )
          {
          if (row != pv )
             {
             factor = a[row][pv] / a[pv][pv];
             for( col = 0 ; col < 3; col++ )
                a[row][col] = a[row][col] - factor * a[pv][col];

             r[row] = r[row] - factor * r[pv];
             }
          }
        }

// Matrix Solution Complete.

     Calibration_Coeffs[0][Port] = r[0];
     Calibration_Coeffs[1][Port] = r[1];
     Calibration_Coeffs[2][Port] = r[2];
     }
}

void Display_Coefficients( void )
{
   char buffer[40], ch;
   int i, Port;
```

```
  _setbkcolor( 1 );
  _settextcolor ( BrightWhite );
  _clearscreen( _GCLEARSCREEN );

  OutlineScreen();
  _settextposition( 3, 0);
  _outtext( LeftSideDivider );

  for ( i = 1; i < 79; i++)
  _outtext( Horizontal );

  _outtext( RightSideDivider );

  for ( i = 4; i < 24; i++ )
      {
      _settextposition( i, 40 );
      _outtext( "³" );
      }
  _settextposition( 3, 40 );
  _outtext( "Ñ" );

  _settextposition( 2, 26);
  _outtext("ESP Calilbration Coefficients");

  for( Port = 0; Port < NumberOfPorts; Port++ )
     {
      if( Port/2*2 == Port )   // Then it be even
         {
         _settextposition( Port/2 + 5, 4 );
         sprintf(buffer,"%2d  %9.5f  %9.5f  %9.5f", Port+1,
             Calibration_Coeffs[0][Port], Calibration_Coeffs[1][Port],
             Calibration_Coeffs[2][Port]);
         _outtext( buffer );
         }
      else
         {
         _settextposition( Port/2 + 5, 42 );
         sprintf(buffer,"%2d  %9.5f  %9.5f  %9.5f", Port+1,
             Calibration_Coeffs[0][Port], Calibration_Coeffs[1][Port],
             Calibration_Coeffs[2][Port]);
         _outtext( buffer );
         }
     }

  while ( !kbhit() )
  {}
  ch = getch();
  if ( ch == 0 )  ch = getch();

}


void Freestream( void )
```

```
{
   int ESC_Flag, i;
   float Voltage, Pressure, Velocity;
   char ch, buffer[40];

   _setbkcolor( 1 );
   _settextcolor ( BrightWhite );

   _clearscreen( _GCLEARSCREEN );
   OutlineScreen();

   _settextcolor ( Green );
   _settextposition( 20, 21 );
   _outtext("Press ESC when Correct Speed is Reached.");
   _settextcolor ( BrightWhite );

   ESC_Flag = 0;

   while( !ESC_Flag )
      {
      ESP_PortNumber = 16;  // Assuming the Total pressure is on Port 16
      Voltage  = 0.000;
      for( i = 0; i < 15; i++ )
         Voltage  = Voltage + Address_and_Sample();

      Voltage  =   Voltage / 15.0;
      Pressure =   Calibration_Coeffs[0][15]
                 + Calibration_Coeffs[1][15] * Voltage
                 + Calibration_Coeffs[2][15] * Voltage * Voltage;
      Velocity = sqrt( 2.0 * fabs(Pressure) * 133.35 / 1.2745 );

      _settextposition( 10, 26 );
      sprintf( buffer, "Dynamic Pressure:  %8.4f torr", Pressure );
      _outtext( buffer );
      _settextposition( 11, 30 );
      sprintf( buffer, "Tunnel Speed: %9.2f m/s", Velocity );
      _outtext( buffer );

      if( kbhit() )
         {
         ch = getch();
         if ( ch == 27 ) ESC_Flag = 1;
         if ( ch == 0 )  ch = getch();
         }
      }

   _clearscreen( _GCLEARSCREEN );
}


float Get_LVDT_Value( void )
{
   int data, i, hi,lo;
```

```
   float voltage, value;
   char Buffer[10];

   outp(0x030B,0x0);     // Disabling Data Acqusition System
   outp(0x0311,0x0);     // Internal Clock

   for ( i=0; i<16; i++)  // Loading Multiplexer Table
      outp(0x0310,0x0);   // Channel 0 is only one being used

   outp(0x0308,0x0);     // Trigger always, internal trigger, minus polarity
   outp(0x031A,0x8);     // Trigger channel 1, IRQ3 on post trigger delay

   outp(0x0307,0x74);    // set clock for 333 usec ( 3000 Hz Sampling )
   outp(0x0305,0x4D);     // Load LSB
   outp(0x0305,0x01);     // Load MSB

   outp(0x0307,0x32);    // Sample 1 channel each burst
   outp(0x0304,0x1);     // Load LSB
   outp(0x0304,0x0);     // Load MSB

   outp(0x0307,0xB2);    // set post trigger delay for 100 samples
   outp(0x0306,0x64);     // Load LSB
   outp(0x0306,0x00);     // Load MSB

   outp(0x0315,0x0);     // Select bank B
   outp(0x031B,0x0);     // reset bank A memory pointer
   outp(0x030D,0x0);     // Start acquisition and post trigger delay

   Delay( 50L );         // Delay For the time of acquisition

   outp(0x0314,0x0);     // select bank A
   outp(0x031B,0x0);     // reset bank A memory pointer

   voltage = 0.0000;

   for ( i=0; i<100; i++)
      {
      lo      = inp(0x031C);
      hi      = inp(0x031C);
      data    = lo + 256 * hi;
      voltage = voltage + (data - 2048) * 0.004883;
      }

   voltage = voltage / 100.000;

   _settextposition( 14, 62);
   sprintf(Buffer,"%6.3f", voltage);
   _outtext( Buffer );
   CursorHome;

   return ( voltage );
}
```

```
void FindStartPosition( void )
{
       float LVDT_Voltage, OldValue;
static float Tol = 0.06;
         int StartFlag;
        char ch;

   StartFlag = 0;

   _settextposition( 12, 21);  // RC Status location
   _outtext("Finding Start ...       ");

   _settextposition( 12, 59);
   _settextcolor( Red + Blinking );
   _outtext("Re-Initializing ...");
   _settextcolor( BrightWhite );

   OldValue = Get_LVDT_Value();

   while( !StartFlag )
      {
      LVDT_Voltage = Get_LVDT_Value();

      if ( LVDT_Voltage > -3.500 )
         outp( 0x0378, FastIdleCommand );
      else
         outp( 0x0378, SlowIdleCommand );

      if ((LVDT_Voltage > StartVoltage-Tol) &&
                                 (LVDT_Voltage < StartVoltage+Tol)  )
         {
         outp( 0x0378, StopCommand );
         if( (LVDT_Voltage-OldValue) >= 0.0000 ) StartFlag = 1;
         if( fabs(LVDT_Voltage-OldValue) < 0.01) StartFlag = 1;  // for
steady
         }

      OldValue = LVDT_Voltage;
      if( kbhit() )
         {
         ch = getch();
         if ( ch == 0 )  ch = getch();
         }
      }
}

void ReadStartPosition( void )
{
  char ch;
  int i, j, RC_Read = 1, Response = 0;

  _setbkcolor( 1 );
  _settextcolor ( BrightWhite );
```

```
_clearscreen( _GCLEARSCREEN );

OutlineScreen();

_settextposition( 10, 15 );
_outtext("Do you wish to manually enter the start LVDT voltage");
_settextposition( 11, 24 );
_outtext("or have the RC board read it now ?");

while( !Response )
   {
   if ( RC_Read )       // Read is selected option
      {
      _settextposition( 15, 30);
      _setbkcolor( 3 );
      _settextcolor( BrightWhite );
      _outtext("Read Now");
      _settextposition( 15, 46);
      _setbkcolor( 1 );
      _settextcolor( Red );
      _outtext("Manual Entry");
      }
   else
     {
      _settextposition( 15, 30);
      _setbkcolor( 1 );
      _settextcolor( Red );
      _outtext("Read Now");
      _settextposition( 15, 46);
      _setbkcolor( 3 );
      _settextcolor( BrightWhite );
      _outtext("Manual Entry");
      }

   while( !kbhit ) {}
   ch = getch();
   if ( ch == 13 ) Response = 1;           // ENTER
   if ( ch == 32 ) RC_Read = !RC_Read; // SPACE-BAR
   if ( ch ==  9 ) RC_Read = !RC_Read; // TAB
   if ( ch ==  0 ) ch = getch();
   }

_setbkcolor( 1 );
_settextcolor ( BrightWhite );
_clearscreen( _GCLEARSCREEN );

if ( RC_Read )
   {
   StartVoltage = 0.00000;
    for ( i = 0; i < 10; i++ )
      StartVoltage += Get_LVDT_Value();

   StartVoltage = StartVoltage/10.0;
```

```
     }
  else
     {
     printf("Enter the LVDT voltage for the start -->\t");
     scanf("%f", &StartVoltage);
     }

}

void Delay ( long Milliseconds )
{
// Attempts to create a Turbo Pascal-like delay for x milliseconds routine
//
  clock_t time_1, time_2;

  time_1 = clock();   // Returns the amount of time in milliseconds
  time_2 = clock();   // since the program has begun.

  while ( (time_2 - time_1) < Milliseconds )
    {
    time_2 = clock();
    }
}

int Round( float RealValue )
{
   int RoundedValue;

   RoundedValue = (int)( RealValue);
   if( (RealValue - (float)(RoundedValue)) > 0.50000 )
           RoundedValue++;

   return( RoundedValue );
}


void BuildScreen( void )
{
   int i;
   char date[40], buffer[90];

   _setbkcolor( 1 );
   _settextcolor ( 15 );
   _clearscreen( _GCLEARSCREEN );

   OutlineScreen();

   _settextposition( 3, 0);
   _outtext( LeftSideDivider );

   for ( i = 1; i < 79; i++)
      _outtext( Horizontal );
```

```
   _outtext( RightSideDivider );

   _settextposition( 2, 17);
   _outtext("P I T C H   U P    S W I N G    M A N A G E R");

   _settextposition( 5, 31);
   GetDate( date );
   _outtext( date );

   _settextposition( 7, 27);
   sprintf(buffer, "DataFile Name: %s", filename);
   _outtext( buffer );

   _settextposition( 9, 30 );
   sprintf(buffer,"Ensemble:      of %d", NumberOfEnsembles);
   _outtext( buffer );

   _settextposition( 12, 6);
   _outtext("System Status:");

   _settextposition( 12, 45 );
   _outtext("Motor Status: ");

   _settextposition( 14, 45);
   _outtext("LVDT Voltage:");

   _settextposition( 15, 45 );
   sprintf( buffer, "Start Voltage:   %6.3f", StartVoltage );
   _outtext( buffer );

}

void OutlineScreen( void )
{
   int i;

   for ( i = 1; i < 80; i++)
      {
      _settextposition( 1, i );
      _outtext( Horizontal );
      _settextposition( 24, i );
      _outtext( Horizontal );
      }

   for ( i = 2; i < 25; i++ )
      {
      _settextposition( i, 0 );
      _outtext( Vertical );
      _settextposition( i, 80 );
      _outtext( Vertical );
      }

   _settextposition(24, 0);
```

```
   _outtext( LowerLeftCorner );

   _settextposition( 24, 80 );
   _outtext( LowerRightCorner );

   _settextposition(1, 0);
   _outtext( TopLeftCorner );

   _settextposition( 1, 80 );
   _outtext( TopRightCorner );

   _settextposition( 24, 24);
   _outtext(" ESM Fluid Mechanics Laboratory ");
}

void GetDate( char date_string[40]  )
{
   struct dosdate_t date;
   char day_year[10];

   _dos_getdate( &date );

   switch ( date.month )
      {
      case 1:
        strcpy( date_string, "January" );
        break;
      case 2:
        strcpy( date_string, "Feburary" );
        break;
      case 3:
        strcpy( date_string, "March" );
        break;
      case 4:
        strcpy( date_string, "April" );
        break;
      case 5:
        strcpy( date_string, "May" );
        break;
      case 6:
        strcpy( date_string, "June" );
        break;
      case 7:
        strcpy( date_string, "July" );
        break;
      case 8:
        strcpy( date_string, "August" );
        break;
      case 9:
        strcpy( date_string, "September" );
        break;
      case 10:
        strcpy( date_string, "October" );
```

```
        break;
      case 11:
        strcpy( date_string, "November" );
        break;
      case 12:
        strcpy( date_string, "December" );
        break;
   }

   strcat( date_string, " ");
   sprintf( day_year,"%u, %4d", date.day, date.year);
   strcat( date_string, day_year);

}

void ShutDownModel60( void )
{
   outp( 0x0378, ShutDownCommand);       // ISA Computer LPT1
   Delay( 50L );
   outp( 0x0378, 0);
}

void  We_Are_Triggered( void )
{
     _settextposition( 12, 21);  // RC Status location
     _outtext("Triggered.             ");
     _settextposition( 12, 59);  // Motor Status
     _outtext("Pitching ...       ");
}

unsigned char HI( unsigned int input)
{
   unsigned char hi_byte;

   hi_byte = (unsigned char)( input >> 8 );
   return( hi_byte );
}
unsigned char LOW( unsigned int input)
{
   unsigned char lo_byte;

   lo_byte = (unsigned char)( (input << 8 ) >> 8 );
   return( lo_byte );
}


*** Listing of Screen.h ********

#define LowerLeftCorner "╚"
#define TopLeftCorner   "╔"

#define LowerRightCorner "╝"
#define TopRightCorner   "╗"
```

```
#define BottomOfDivider "╧"
#define TopOfDivider    "╤"

#define LeftSideDivider "╟"
#define RightSideDivider "╢"

#define Horizontal "─"
#define Vertical   "│"

#define CenterCross "┼"

#define Green 10
#define Red 12
#define BrightWhite 15
#define Blinking 16
```

APPENDIX C: "*MOTORMAN*" CODE LISTING

This appendix contains the code listing for the "Motorman" program which ran on the IBM PS/2 Model 60 while taking data at the ESM Wind Tunnel facility. Motorman receives commands from the ISA PC via its parallel port, which is set by Motorman into an input mode. Based on the command received, Motorman varies the voltage output from the D/A portion of the Data Translation DT2905 board installed in the computer. The voltage is an input to the Morse motor controller, which controls the pitch motor. The level of the voltage determines the speed of the motor. A listing of "screen.h" is included at the end of Appendix B.

```
//      MOTORMAN.C   The Pitch Up Swing Motor Manager!
//         (c) 1994 Norman W. Schaeffler
//
//  Includes: Communication with Model 60 via Parallel Port
//

// C includes
#include <graph.h>
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <stdlib.h>

// Project Includes
#include "screen.h"

// Defines
#define CursorHome   _settextposition( 23, 2 )
#define Base_Freq 10000000
#define SlowIdle  1.070
#define FastIdle  2.250
#define Stop      0.000
#define ESC       "\x1B"

void Initialize_DA_Channel0( float );
```

```
void Initialize_DA_Channel1( float );
void DA_Channel0( float );
unsigned int ConvertVoltage( float );
void DA_Array( unsigned int [], int );
void StartMotor( void );
void Delay ( long Milliseconds );
void BuildScreen( void );
void GetDate( char date_string[40]  );
void BiDirectionalParallelPort_On( void );
void BiDirectionalParallelPort_Off( void );
unsigned int mask( char binary[] );


// Global Variables
unsigned int Base = 0x3000;
int Initialization_Flag = 0;
float Real_Frequency;

void main( int argc, char *argv[] )
{
   int i, samples, samples_over_two, ParallelValue, ExitFlag;
   long TenVoltTime;
   int PrevCommand;
   char ch, parallel[4], buffer[40];
   float Voltage, ramp_duration, Channel1;
   float VoltStart, VoltEnd;
   float A, B, delta_t, Time;
   unsigned int DA_Values[3100];
   clock_t time_1, time_2;

   _clearscreen( _GCLEARSCREEN );

   if( argc != 3 )
      {
      printf("DC Motor Manager                                (c)1994 N.W.
Schaeffler\n");
      printf("Usage    C:>motorman delay D/A#1\nWhere delay is in
microseconds ( about 150)\n");
      printf("and D/A#1 is the voltage to be output on channel 1\n");
      exit(0);
      }
   else
      {
      printf("Arguments: %s %s\n", argv[1], argv[2]);
      TenVoltTime = atol( argv[1] );
      Channel1    = atof( argv[2] );
      printf("Arguments: %ld %f\n", TenVoltTime, Channel1);
      }

   Initialize_DA_Channel1( Channel1 );
   Initialize_DA_Channel0( 0.0000 );

// VoltStart and VoltEnd are in terms of motor voltages, not D/A Voltages
```

```
   ramp_duration =  0.750; // Seconds
   VoltStart     = 10.000;  // Voltage at begining of ramp
   VoltEnd       =  0.000;  // Voltage at end

   BiDirectionalParallelPort_On();
   StartMotor();

   samples = (int)( Real_Frequency * ramp_duration );

// Uses  V = A * time^2 + B as the functional form of the ramp

   B =  VoltEnd;
   A = (VoltStart-B )/(ramp_duration * ramp_duration);
   delta_t = 1/Real_Frequency;

   for ( i = 0; i < samples; i++ )
      {
       Time = ramp_duration - (i-1) * delta_t;
       Voltage = A * Time * Time + B ;
       DA_Values[i] = ConvertVoltage( Voltage );
      }

   printf("Ramp Values Computed.\n");
   printf("\n\n\nS T A T I S T I C S\n\n");
   printf("Frequency --> %f Hz\nNumber of Samples --> %d\nRamp Duration -->
%f seconds\n", Real_Frequency, samples, ramp_duration );
   printf("\nReady to switch to ISA Control\n\nHit any key to Continue
...\n");

   while( !kbhit() )  {}
   ch = getch();
   if ( ch == 0 ) ch = getch();

   BuildScreen();
   _settextposition( 18, 27);
   sprintf(buffer,"Trigger Value: %7.4f Volts", Channel1);
   _outtext( buffer );

   ExitFlag = 0;
   PrevCommand = 0;

   while( !ExitFlag )
      {
       ParallelValue = inp( 0x03BC );

       if ( ParallelValue != PrevCommand )
          {
          PrevCommand = ParallelValue;
          switch( ParallelValue )
            {
            case 0:
                Initialize_DA_Channel1( Channel1 );
```

```
        Initialize_DA_Channel0( 0.0000 );
        DA_Channel0( Stop );
        _settextposition(12,43);
        _outtext("   0");
        _settextposition(14,37);
        _outtext("Motor Stopped ...      ");
        break;

    case 10:
        DA_Channel0( SlowIdle );
        _settextposition(12,43);
        _outtext("  10");
        _settextposition(14,37);
        _outtext("Idle Motor Slow ...    ");
        break;

    case 20:
        DA_Channel0( FastIdle );
        _settextposition(12,43);
        _outtext("  20");
        _settextposition(14,37);
        _outtext("Idle Motor Fast ...    ");
        break;

    case 65:
        _settextposition(12,43);
        _outtext("  65");
        _settextposition(14,37);
        _outtext("Prepared to Pitch ...  ");

        DA_Channel0( 10.00 );
        while( (inp( 0x03BC )) != 128 )
           {}                                  // Trigger Loop
        Delay( TenVoltTime );
        DA_Array( DA_Values, samples );
        break;

    case 128:
        _settextposition(12,43);
        _outtext("  65");
        _settextposition(14,37);
        _outtext("Trigger Received.      ");
        break;

    case 255:
        DA_Channel0( Stop );
        _settextposition(12,43);
        _outtext(" 255");
        _settextposition(14,37);
        _outtext("End of Run Received.   ");
        break;

    default:
```

```
                DA_Channel0( Stop );
                _settextposition(12,43);
                sprintf( parallel, "%4d", ParallelValue );
                _outtext( parallel );
                _settextposition(14,37);
                _outtext("Unrecognized Command   ");
                break;
              }
            }

        if ( kbhit() )
            {
            ch = getch();
            if ( ch == 27 ) ExitFlag = 1;
            if ( ch == 0 ) ch = getch();
            }
       }

   _clearscreen( _GCLEARSCREEN );
   DA_Channel0( Stop );
   BiDirectionalParallelPort_Off();

} // That's All Folks !!!!!

void Initialize_DA_Channel0( float voltage )
{
   unsigned int PreScalar, Counter, clock;
   unsigned int integer_voltage, Channel;
   int i, two_to_the_PreScalar;

//   printf("\nInitializing D/A Channel 0 ... ");
   outpw( Base + 0xE, inpw( Base + 0xE ) | 0x8000 );
   outpw( Base + 0x4, inpw( Base +  4 ) & 0xFFF3 );
   outpw( Base + 0x6, inpw( Base +  6 ) & 0x0100 );

   Channel = inpw( Base + 4) | 0x0100 & 0xFDFF;
   outpw( Base + 0x4, Channel );

   integer_voltage = (int)( (voltage + 5.00) * 4096 / 10.000 );
   if( integer_voltage > 4095 ) integer_voltage = 4095;
   if( integer_voltage < 0 ) integer_voltage = 0; // This line may be
unneccesary
   outpw( Base + 0x8, integer_voltage);

   PreScalar = 10;  // 8 = 250 Hz
   Counter = 156;
   two_to_the_PreScalar = 1;
   for ( i = 0; i < PreScalar; i++ )
      two_to_the_PreScalar *= 2;

   Real_Frequency = Base_Freq / (float)(two_to_the_PreScalar)
                                                    / (float)(Counter);
```

```
   clock = ((inpw( Base + 0xE ) & 0x8000) + 0x4000) | ( PreScalar << 8 ) |
Counter;
   outpw( Base + 0xE, clock );
   Initialization_Flag = 1;
//   printf(" Done.\n\n");
}

void Initialize_DA_Channel1( float voltage )
{
   unsigned int PreScalar, Counter, clock;
   unsigned int integer_voltage, Channel;
   int i, two_to_the_PreScalar;

//   printf("\nInitializing D/A Channel 1 ... ");
   outpw( Base + 0xE, inpw( Base + 0xE ) | 0x8000 );
   outpw( Base + 0x4, inpw( Base +  4 ) & 0xFFF3 );
   outpw( Base + 0x6, inpw( Base +  6 ) & 0x0100 );

   Channel = inpw( Base + 4) | 0x0300;
   outpw( Base + 0x4, Channel );

   integer_voltage = (int)( (voltage + 5.00) * 4096 / 10.000 );
   if( integer_voltage > 4095 ) integer_voltage = 4095;
   if( integer_voltage < 0 ) integer_voltage = 0; // This line may be
unneccesary
   outpw( Base + 0x8, integer_voltage);

   PreScalar = 10;  // 8 = 250 Hz
   Counter = 156;
   two_to_the_PreScalar = 1;
   for ( i = 0; i < PreScalar; i++ )
      two_to_the_PreScalar *= 2;

   Real_Frequency = Base_Freq / (float)(two_to_the_PreScalar)
                                                    / (float)(Counter);
   clock = ((inpw( Base + 0xE ) & 0x8000) + 0x4000) | ( PreScalar << 8 ) |
Counter;
   outpw( Base + 0xE, clock );
//   printf(" Done.\n\n");
}

void DA_Channel0( float MotorVoltage )
{
   unsigned int integer_voltage;
   int i;

   if( !Initialization_Flag )
       {
       printf("D/A Subsystem not initialized. Fatal Error, my friend\n");
       return;
       }

   integer_voltage = ConvertVoltage( MotorVoltage );
```

```
   outpw( Base + 0x8, integer_voltage);

}

unsigned int ConvertVoltage( float MotorVoltage )
{
// This routine is to convert Motor voltages into the corresponding
// D/A Voltage and then convert this voltage to the corresponding
// integer voltage.

   unsigned int integer_voltage;
   float DA_Voltage, a0, a1, a2, a3;

   if( MotorVoltage > 9.75 ) MotorVoltage = 9.75;  // Maximum Safe Voltage

   a0 =  1.23338;
   a1 =  0.247886;
   a2 = -0.0201841;
   a3 =  0.000976;

   DA_Voltage =   a3 * MotorVoltage * MotorVoltage * MotorVoltage
               + a2 * MotorVoltage * MotorVoltage
               + a1 * MotorVoltage
               + a0;

   integer_voltage = (unsigned int)( (DA_Voltage + 5.00) * 4096 / 10.000 );
   if( integer_voltage > 4095 ) integer_voltage = 4095;
   if( integer_voltage < 0 ) integer_voltage = 0; // This line may be
unneccesary

   return( integer_voltage );

}

void DA_Array( unsigned int values[], int Number_of_Conversions )
{
   int i;

   if( !Initialization_Flag )
      {
      printf("D/A Subsystem not initialized. Fatal Error, my friend\n");
      return;
      }

   for( i = 0; i < Number_of_Conversions; i++)
      {
      outpw( Base + 0x08, values[i] );
      while( (inpw( Base + 0x4 ) & 2048) != 2048 ) {};
      }
}

void StartMotor( void )
{
```

```
   char ch;

   DA_Channel0( 2.000 );
   printf("\nStart Motor Now. Press any key when sucessfull. \n\n");

   while( !kbhit() )  {}
   ch = getch();
   if ( ch == 0 ) ch = getch();

   DA_Channel0( 0.000 );
}

void Delay ( long Milliseconds )
{
// Attempts to create a Turbo Pascal-like delay for x milliseconds routine
//
  clock_t time_1, time_2;

  time_1 = clock();   // Returns the amount of time in milliseconds
  time_2 = clock();   // since the program has begun.

  while ( (time_2 - time_1) < Milliseconds )
    {
    time_2 = clock();
    }
}

void BuildScreen( void )
{
   int i;
   char date[40];

   _setbkcolor( 1 );
   _settextcolor ( 15 );
   _clearscreen( _GCLEARSCREEN );

   for ( i = 1; i < 80; i++)
      {
      _settextposition( 2, i );
      _outtext( Horizontal );
      _settextposition( 24, i );
      _outtext( Horizontal );
      }

   for ( i = 3; i < 25; i++ )
      {
      _settextposition( i, 0 );
      _outtext( Vertical );
      _settextposition( i, 80 );
      _outtext( Vertical );
      }

   _settextposition(24, 0);
```

```
   _outtext( LowerLeftCorner );

   _settextposition( 24, 80 );
   _outtext( LowerRightCorner );

   _settextposition(2, 0);
   _outtext( TopLeftCorner );

   _settextposition( 2, 80 );
   _outtext( TopRightCorner );

   _settextposition( 4, 0);
   _outtext( LeftSideDivider );

   for ( i = 1; i < 79; i++)
      _outtext( Horizontal );

   _outtext( RightSideDivider );

   _settextposition( 3, 22);
   _outtext("D C    M O T O R    M A N A G E R");

   _settextposition( 24, 24);
   _outtext(" ESM Fluid Mechanics Laboratory ");

   _settextposition( 6, 28);
   GetDate( date );
   _outtext( date );
   _settextposition( 12, 20 );
   _outtext("Last Received Command: ");
   _settextposition( 14, 20 );
   _outtext("Interpretation: ");
}

void GetDate( char date_string[40]  )
{
   struct dosdate_t date;
   char day_year[10];

   _dos_getdate( &date );

   switch ( date.month )
      {
      case 1:
        strcpy( date_string, "January" );
        break;
      case 2:
        strcpy( date_string, "Feburary" );
        break;
      case 3:
        strcpy( date_string, "March" );
        break;
      case 4:
```

```
        strcpy( date_string, "April" );
        break;
      case 5:
        strcpy( date_string, "May" );
        break;
      case 6:
        strcpy( date_string, "June" );
        break;
      case 7:
        strcpy( date_string, "July" );
        break;
      case 8:
        strcpy( date_string, "August" );
        break;
      case 9:
        strcpy( date_string, "September" );
        break;
      case 10:
        strcpy( date_string, "October" );
        break;
      case 11:
        strcpy( date_string, "November" );
        break;
      case 12:
        strcpy( date_string, "December" );
        break;
   }

   strcat( date_string, " ");
   sprintf( day_year,"%u, %4d", date.day, date.year);
   strcat( date_string, day_year);

}

void BiDirectionalParallelPort_On( void )
{
// IMPORTANT!!!!!! IMPORTANT!!!!!!
// This routine is for a PS/2 with an ENHANCED parallel port. The code
// seeks to setup bidirectional status for the port and then sets up the
// port to read.
//
   unsigned SetupRegister, Enabled, POS_Register2, ControlPort;

   Enabled = inp( 0x0094 );
   SetupRegister = Enabled & mask( "01111111" );
   printf("\nEntering SetUp.\nEnabling BiDirectional Parallel Port ...");
   outp( 0x0094, SetupRegister );     // Places system into setup mode

   POS_Register2 = inp( 0x0102 );
   POS_Register2 = POS_Register2 & mask( "01111111" );
   outp( 0x0102, POS_Register2 );         // Allows BiDirectional

   outp( 0x0094, Enabled );       // Takes system out of setup mode
```

```
   ControlPort = inp( 0x03BE );   // Read current values at Control Port
   ControlPort = ControlPort | mask("00100000");
   outp( 0x03BE, ControlPort );   // Places Port in read mode
   printf(" Finished\n");

}

void BiDirectionalParallelPort_Off( void )
{
// IMPORTANT!!!!!! IMPORTANT!!!!!!
// This routine is for a PS/2 with an ENHANCED parallel port. The code
// disables bidirectional status for the port, restoring the default config.
//
   unsigned SetupRegister, Enabled, POS_Register2, ControlPort;

   Enabled = inp( 0x0094 );
   SetupRegister = Enabled & mask( "01111111" );
   printf("Entering SetUp.\nDisabling BiDirectional Parallel Port ...");
   outp( 0x0094, SetupRegister );     // Places system into setup mode

   POS_Register2 = inp( 0x0102 );
   POS_Register2 = POS_Register2 | mask( "10000000" );
   outp( 0x0102, POS_Register2 );         // Disables BiDirectional

   outp( 0x0094, Enabled );       // Takes system out of setup mode
   printf(" Finished \n");

}

unsigned int mask( char binary[] )
{
   int   i;
   unsigned long power, decimal;

   decimal = 0;
   power = 1;

   for ( i = 0; i < 8; i++ )
      {
      if( binary[7-i] == '1' )
         decimal += power;
      power = power * 2;
      }

   return ( decimal );
}
```

APPENDIX D: "*PRESSMAN*" CODE LISTING

This appendix contains the code listing for the "Pressman" program which was the data acquisition software for use at the Virginia Tech Stability Wind Tunnel facility. This program was used to acquire the surface pressure measurements from the ESP when the DyPPiR was being used. The code contains provisions to calibrate the ESP and to acquire data from it upon receipt of a trigger. This trigger was provided by the DyPPiR control computer and indicated that the motion had begun. Data from the ESP was acquired from the ESPIO-1 board, data from the position feedback sensors of the DyPPiR was acquired from the RC-ISC16 data acquisition board. The version listed here is from an entry into the Stability Tunnel after the GMF file format was put into use. A ComputerBoards DAS08 was also installed into the computer to count the clock train from the DyPPiR control computer. The DAS08 was programmed, via Pressman, to issue a trigger upon terminal count. This trigger was used to deploy the flaps. The option to get additional trigger signals from the DyPPiR computer was disable when the GMF file format was instituted.

```
//      PRESSMAN.C   The DyPPiR Unsteady Pressure Acquisition Manager!
//         (c) 1994-1996 Norman W. Schaeffler
//

// C includes
#include <graph.h>
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <math.h>

// Defines
```

```
#define LowerLeftCorner "╚"
#define TopLeftCorner   "╔"

#define LowerRightCorner "╝"
#define TopRightCorner   "╗"

#define BottomOfDivider "╧"
#define TopOfDivider    "╤"

#define LeftSideDivider "╟"
#define RightSideDivider "╢"

#define Horizontal "─"
#define Vertical   "│"

#define CenterCross "┼"

#define Green 10
#define Red 12
#define BrightWhite 15
#define Blinking 16

#define Internal 0
#define External 1

#define NoCursor  0x2000
#define Underline 0x0707
#define CursorHome   _settextposition( 23, 2 )

#define byte unsigned char
#define word unsigned  int

// Function Protoypes
//     ESP Related
          void  SetUpESP( void );
          void  ESP_TestSuite( void );
         float  Address_and_Sample( void );
          void  AcquirePressuresOnTrigger( void );
          void  CalibrateESP( void );
          void  PerformESPCalibration( void );
          void  Compute_Coefficients( void );
          void  SaveESPData( void );
          void  Freestream( void );
//    RC Related
          void  AcquireRCDataOnTrigger( void );
          void  SaveRCData( void );
//  Utility
          void SetUpCounter( unsigned int );
          void  Delay( long );
          void  BuildScreen( void );
          void  OutlineScreen( void );
          void  GetParameters( void );
```

```
          void  GetDate( char date_string[40] );
unsigned  char  HI( unsigned int );
unsigned  char  LOW( unsigned int );
           int  Round( float );

// Structures
struct ESP_Info
   {
    char date[40];
    int  NumberOfEnsembles;
    int  NumberOfChannels;
    int  NumberOfPorts;
    int  SamplingRate;
    int  Number_Of_Samples;
    int  ESP_SamplingRate;
    int  Number_Of_ESP_Samples;
   };

// Global Variables
       struct ESP_Info info;
          int NumberOfCalPoints, Number_Of_Samples, NumberOfChannels;
          int BufferSize, SamplingRate, DurationOfSampling;
         char filename[12], date[40];
         char CommandString1[60];
         FILE *DataFile;
          int ESP_PortNumber, ESP_OperationMode, NumberOfEnsembles;
          int NumberClockPulses, Number_Of_ESP_Samples, NumberOfPorts;
        float ESP_SamplingRate, ESP_BurstRate, Time_Of_ESP_Sampling;
       double Voltage[10][32], Pressure[10], Calibration_Coeffs[3][32];
          int ESP_32_Flag;
 unsigned int NumberOfClockPulses;

          int RC_Data[1500];     // Must Match Number_Of_Samples
          int hi[310][32],lo[310][32];  // = ESP_SamplingRate * ESP_Duration

void main( void )
{
   char buffer[12], ch;
   int i, j;

   _clearscreen( _GCLEARSCREEN );
   _settextcursor( NoCursor );

   GetParameters();
   ESP_TestSuite();
   CalibrateESP();

// RC Stuff ***** MODIFY ARRAY BOUNDS TOO IF YOU CHANGE THESE *****
   DurationOfSampling = 1.50;    // In Seconds
   SamplingRate       = 500;     // In Hz
   NumberOfChannels   = 2;
   Number_Of_Samples  = NumberOfChannels * SamplingRate * DurationOfSampling;
```

```
// ESP Stuff ***** MODIFY ARRAY BOUNDS TOO IF YOU CHANGE THESE *****
   ESP_32_Flag          = 0;     // 1 if 32 channel, 0 if 16 channel
   ESP_SamplingRate     = 200.0; // Hz.
   Time_Of_ESP_Sampling = 1.500; // Seconds


   if ( (DataFile = fopen(filename,"wb")) == NULL )
     {
     _clearscreen( _GCLEARSCREEN );
     printf("Error opening file %s\nProgram Aborted\n", filename);
     exit(0);
     }

   SetUpESP();

// Fill and Save info structure and Calibration Coeffs.

   GetDate( info.date );
   info.NumberOfEnsembles     = NumberOfEnsembles;
   info.NumberOfChannels      = NumberOfChannels;
   info.NumberOfPorts         = NumberOfPorts;
   info.SamplingRate          = SamplingRate;
   info.Number_Of_Samples     = Number_Of_Samples;
   info.ESP_SamplingRate      = ESP_SamplingRate;
   info.Number_Of_ESP_Samples = Number_Of_ESP_Samples;

   fwrite( &info, sizeof( info ), 1, DataFile );
   fwrite( Calibration_Coeffs, sizeof( Calibration_Coeffs ), 1, DataFile);

   Freestream();

   BuildScreen();


   for( i = 0; i < NumberOfEnsembles;  i++)
     {
     _settextposition( 14, 40 );
     sprintf(buffer,"%4d", i+1);   // Ensemble Number
     _outtext( buffer );

     AcquireRCDataOnTrigger();
     SetUpCounter( NumberOfClockPulses );
     AcquirePressuresOnTrigger();

     SaveESPData();
     SaveRCData();

     _settextposition( 16, 41);
     _outtext("Paused for Re-Initializing.       ");
     Delay(5500L);
     SetUpCounter( NumberOfClockPulses );
     Delay(10000L);
     }
```

```
   fclose( DataFile );
   _settextcursor( Underline );
   _setbkcolor( 0 );
   _settextcolor ( BrightWhite );
   _clearscreen( _GCLEARSCREEN );

   strcpy( CommandString1, "ATTRIB *.RAW + R");
   execl("COMMAND.COM", " ", CommandString1, " ", NULL);

   SetUpCounter( NumberOfClockPulses );

// Get the hell out of dodge!
}

void GetParameters( void )
{
   char ch;

   _settextcolor(15);
   _setbkcolor(0);
   _clearscreen( _GCLEARSCREEN );

   printf("\n\n\tEnter the number of ensembles -->\t");
   scanf("%d", &NumberOfEnsembles);

   printf("\tEnter the number of ESP channels to be used -->\t");
   scanf("%d", &NumberOfPorts);

   printf("\tEnter the filename for the data  -->\t");
   scanf("%s", &filename);

   printf("\tEnter the Number of Clock Pulses for Flap Delay -->\t");
   scanf("%d", &NumberOfClockPulses);

}

void SetUpESP( void )
{
//
// ESP Operation Modes
// 160 - Multiple Readings, internal oscillator, software trigger
//  32 - Multiple Readings, internal oscillator, external trigger
//  64 - Multiple Readings, external oscillator
//
// ESP_SamplingRate is the frequency of the between bursts
// ESP_BurstRate is the frequency at which the ESP switches from port to
// port within a burst.
//
//
   float Exponent_Real, clock_freq, sample_freq;
   int   Exponent;
   float NumberClockPulsesAsFloat;
```

```
   char ch;

   ESP_OperationMode    =      32; //
   ESP_BurstRate        = 15625.0; // Hz.

   Exponent_Real = log( 500000.0 / ESP_BurstRate )/ log ( 2.0 );
   Exponent      = Round(Exponent_Real);

   clock_freq  = 1000000.0 / pow( 2.0, Exponent );
   sample_freq = clock_freq / 2.0;
   ESP_OperationMode += Exponent;

   NumberClockPulsesAsFloat = (1.0/ESP_SamplingRate-(NumberOfPorts-
1)/sample_freq)
                              * clock_freq;

   NumberClockPulses = Round( NumberClockPulsesAsFloat );


   ESP_SamplingRate = 1.0/(NumberClockPulses/2.0/sample_freq
                                          + (NumberOfPorts-1)/sample_freq);
   Number_Of_ESP_Samples = (int)( ESP_SamplingRate * Time_Of_ESP_Sampling );

}

void AcquireRCDataOnTrigger( void )
{
   int i, NumberOfMicroSec;
   float voltage, value, DeltaTime;

   DeltaTime = 1.0 / SamplingRate;
   NumberOfMicroSec = (int)( DeltaTime * 1000000.0 );

   outp(0x030B,0x0);     // Disabling Data Acqusition System

   for ( i=0; i<8; i++)  // Loading Multiplexer Table
      {
      outp(0x0310,0x0);   // Channel 0 - 1 are being used
      outp(0x0310,0x1);   // NEEDS to be REWRITTEN to automatically
      }                   // adjust to the number of channels

   outp(0x031A,0x08);   // Trigger channel 1, IRQ3 on post trigger delay
   outp(0x0311,0x00);   // Select Internal Clock
   outp(0x0308,0x0B);   // Trigger slope, external trigger, positive polarity

   outp(0x0307,0x74);     // set clock
   outp(0x0305, LOW(NumberOfMicroSec) );     // Load LSB
   outp(0x0305,  HI(NumberOfMicroSec) );     // Load MSB

   outp(0x0307,0x32);      // Set Number of Channels for each burst
   outp(0x0304,LOW(NumberOfChannels) );     // Load LSB
   outp(0x0304, HI(NumberOfChannels) );     // Load MSB
```

```
   outp(0x0307,0xB2);     // set post trigger delay for number of samples
   outp(0x0306, LOW(Number_Of_Samples) );     // Load LSB
   outp(0x0306,  HI(Number_Of_Samples) );     // Load MSB

   outp(0x0315,0x00);      // Select bank B
   outp(0x031B,0x03);      // reset bank A memory pointer for 2048 samples
   BufferSize = 2048;

   outp(0x030A,0x0);      // Enable Data Acquisition
   outp(0x030C,0x0);      // Enable Trigger Logic

   _settextposition( 16, 41);
   _outtext("Waiting For Trigger ...             ");

}

void SaveRCData( void )
{
   int DataStart, data, i, hi,lo;
   float voltage, delta_time;

   _settextposition( 16, 41);
   _outtext("Saving Data ...                     ");

   DataStart  =  BufferSize - Number_Of_Samples - 1;
   delta_time = 1.0 / SamplingRate;

   outp(0x0314,0x03);     // select bank A

   for ( i=0; i< BufferSize; i++)
      {
      lo      = inp(0x031C);
      hi      = inp(0x031C);
      data    = lo + 256 * hi;
      if( i >= DataStart )
        RC_Data[i-DataStart] = data;
      }

   fwrite( RC_Data, sizeof( RC_Data ), 1, DataFile );
}

float Address_and_Sample( void )
{
   float ESP_Voltage;
   int OldRead, NewRead, Samples, high, low, i;

   outp( 0x3E6, 165 );     // Set up for Internal Oscillator,
                           // Software Trigger, and Sampling Rate of
                           // about 15 kHz ( Othon uses 170 --> 1kHz )

   outp( 0x3E5, ESP_PortNumber - 1);  // address ESP and initialize A/D

   Delay( 50L );
```

```
   ESP_Voltage = 0.0000;
   for ( i = 0; i < 200; i++)
      {
      OldRead = 0;
      Samples = 0;

      while( Samples < 1 )
         {
         NewRead = inp( 0x03E7 );
         NewRead = NewRead & 0x01;

         if ( (OldRead == 1) && (NewRead == 0 ))
            {
            high = inp( 0x03E3 );
            low  = inp( 0x03E2 );
            Samples = Samples + 1;
            ESP_Voltage += ( 16 * high + low/16) * 10.0 / 4096 - 5.0;
            }

         OldRead = NewRead;
         }
      }

   ESP_Voltage = ESP_Voltage/200;
   ESP_Voltage = ESP_Voltage + 0.012 + 0.00346478 * ( 4.064 - ESP_Voltage );
   return( ESP_Voltage );
}

void AcquirePressuresOnTrigger( void )
{
   int OldRead, NewRead, Samples, j, Sample, Port;

   outp( 0x3E6, ESP_OperationMode );     // Set mode and frequency
//
// See older versions of the ESP code for use of the external trigger built
// into the ESP board itself
//
   while ( (inp(0x0309) & 0x0001) == 0 ) // This waits for the Trigger Status
         {  }                            // Bit on the RC board to be set.

   _settextposition( 16, 41);
   _outtext("Triggered ...                      ");

   OldRead = 0;
   for( Sample = 0; Sample < Number_Of_ESP_Samples; Sample++ )
      {
      for( Port = 0; Port < NumberOfPorts; Port++)
         {
         outp( 0x3E5, Port );  // Address ESP
         Samples = 0;
         while( Samples < 2 )
           {
           NewRead = inp( 0x03E7 );
```

```
            NewRead = NewRead & 0x01;

            if ( (OldRead == 1) && (NewRead == 0 ))
               {
               hi[Sample][Port] = inp( 0x03E3 );
               lo[Sample][Port] = inp( 0x03E2 );
               Samples = Samples + 1;
               }

            OldRead = NewRead;
            }
         }

      for( j = 0; j < NumberClockPulses - 2; j++) // Dry sample till begining
         {                                        // of next sample run.
         Samples = 0;
         while( Samples < 1 )
            {
            NewRead = inp( 0x03E7 );
            NewRead = NewRead & 0x01;

            if ( (OldRead == 1) && (NewRead == 0 ))
               Samples = Samples + 1;

            OldRead = NewRead;
            }
         }
      }
}

void SaveESPData( void )
{
   fwrite( hi, sizeof( hi ), 1, DataFile );
   fwrite( lo, sizeof( lo ), 1, DataFile );
}

void ESP_TestSuite( void )
{
   int ESC_Flag = 0, MaxPortNumber;
   float PortVoltage;
   char ch, buffer[12];

   _setbkcolor( 1 );
   _settextcolor ( BrightWhite );

   _clearscreen( _GCLEARSCREEN );
   OutlineScreen();

   _settextposition( 2, 24);
   _outtext("E S P      T E S T     S U I T E");
   _settextposition( 4, 15);
   _outtext("Port");
   _settextposition( 4, 47);
```

```
   _outtext("Port");
   _settextposition( 4, 25);
   _outtext("Voltage");
   _settextposition( 4, 57);
   _outtext("Voltage");
   _settextcolor ( Green + Blinking );
   _settextposition( 22, 29 );
   _outtext("Press ESC to Continue");
   _settextcolor ( BrightWhite );

   ESP_PortNumber = 1;
   while( !ESC_Flag )
      {
      PortVoltage = Address_and_Sample();
      if( ESP_PortNumber/2*2 == ESP_PortNumber )   // Then it be even
         {
         _settextposition( ESP_PortNumber/2 + 4, 48 );
         sprintf( buffer, "%2d  ---> %6.3f", ESP_PortNumber, PortVoltage );
         _outtext( buffer );
         }
      else
         {
         _settextposition( ESP_PortNumber/2 + 5, 16 );
         sprintf( buffer, "%2d  ---> %6.3f", ESP_PortNumber, PortVoltage );
         _outtext( buffer );
         }
      ESP_PortNumber++;

      if( ESP_32_Flag == 1 )  // If true this is a 32 channel ESP
         MaxPortNumber = 32;
      else              //  If false this is a 16 Channel ESP
         MaxPortNumber = 16;

      if( ESP_PortNumber > MaxPortNumber ) ESP_PortNumber = 1;

      if( kbhit() )
         {
         ch = getch();
         if ( ch == 27 ) ESC_Flag = 1;
         if ( ch == 0 )  ch = getch();
         }
      }

   _clearscreen( _GCLEARSCREEN );
}

void CalibrateESP( void )
{
  int Port, i, j, Calibrate = 1, Response = 0;
  char buffer[50], tmpname[14], ch;
  float input;
  struct ESP_Info tmp;
  FILE *coeffs;
```

```
_setbkcolor( 1 );
_settextcolor ( BrightWhite );
_clearscreen( _GCLEARSCREEN );

OutlineScreen();

_settextposition( 10, 10 );
_outtext("Do you wish to calibrate the ESP or read in the coefficients");
_settextposition( 11, 10 );
_outtext("from an existing file?");

while( !Response )
   {
   if ( Calibrate )       // Calibrate is selected option
      {
      _settextposition( 15, 30);
      _setbkcolor( 3 );
      _settextcolor( BrightWhite );
      _outtext("Calibrate");
      _settextposition( 15, 46);
      _setbkcolor( 1 );
      _settextcolor( Red );
      _outtext("Read");
      }
   else
     {
      _settextposition( 15, 30);
      _setbkcolor( 1 );
      _settextcolor( Red );
      _outtext("Calibrate");
      _settextposition( 15, 46);
      _setbkcolor( 3 );
      _settextcolor( BrightWhite );
      _outtext("Read");
      }

   while( !kbhit ) {}
   ch = getch();
   if ( ch == 13 ) Response = 1;           // ENTER
   if ( ch == 32 ) Calibrate = !Calibrate; // SPACE-BAR
   if ( ch ==  9 ) Calibrate = !Calibrate; // TAB
   if ( ch ==  0 ) ch = getch();
   }

 if ( Calibrate )
    PerformESPCalibration();
 else
   {
   _setbkcolor( 0 );
   _settextcolor ( BrightWhite );
   _clearscreen( _GCLEARSCREEN );
   printf("\n\n\tEnter the filename for the data  -->\t");
```

```
    scanf("%s", &tmpname);

    if ( (coeffs = fopen(tmpname,"rb")) == NULL )
       {
       _clearscreen( _GCLEARSCREEN );
       printf("Error opening file %s\nProgram Aborted\n", tmpname);
       exit(0);
       }

    fread( &tmp, sizeof( tmp ), 1, coeffs );
    fread( Calibration_Coeffs, sizeof( Calibration_Coeffs ), 1, coeffs);
    fclose( coeffs );
    }

_setbkcolor( 1 );
_settextcolor ( BrightWhite );
_clearscreen( _GCLEARSCREEN );

OutlineScreen();
_settextposition( 3, 0);
_outtext( LeftSideDivider );

for ( i = 1; i < 79; i++)
_outtext( Horizontal );

_outtext( RightSideDivider );

for ( i = 4; i < 24; i++ )
    {
    _settextposition( i, 40 );
    _outtext( "³" );
    }
_settextposition( 3, 40 );
_outtext( "Ñ" );

_settextposition( 2, 26);
_outtext("ESP Calilbration Coefficients");

for( Port = 0; Port < NumberOfPorts; Port++ )
   {
    if( Port/2*2 == Port )   // Then it be even
       {
       _settextposition( Port/2 + 5, 4 );
       sprintf(buffer,"%2d  %9.5f  %9.5f  %9.5f", Port+1,
           Calibration_Coeffs[0][Port], Calibration_Coeffs[1][Port],
           Calibration_Coeffs[2][Port]);
       _outtext( buffer );
       }
    else
       {
       _settextposition( Port/2 + 5, 42 );
       sprintf(buffer,"%2d  %9.5f  %9.5f  %9.5f", Port+1,
           Calibration_Coeffs[0][Port], Calibration_Coeffs[1][Port],
```

```
            Calibration_Coeffs[2][Port]);
         _outtext( buffer );
         }
      }

  while ( !kbhit() )
  {}
  ch = getch();
  if ( ch == 0 )  ch = getch();

  if ( Calibrate )
     {
     _settextcolor ( Green + Blinking );
     _clearscreen( _GCLEARSCREEN );
     _settextposition( 12, 20 );
     _outtext("Apply 80 psi to C2 to set ESP in Run Mode");

     while ( !kbhit() )
        {}
     ch = getch();
     if ( ch == 0 )  ch = getch();
     }


  _settextcolor ( BrightWhite );
}

void PerformESPCalibration( void )
{
  int Port, i, j;
  char buffer[50], ch;
  float input;

  _settextcolor ( Green + Blinking );
  _clearscreen( _GCLEARSCREEN );
  _settextposition( 12, 20 );
  _outtext("Apply 80 psi to C1 to set ESP in Calibration Mode");

  while ( !kbhit() )
  {}
  ch = getch();
  if ( ch == 0 )  ch = getch();

  _setbkcolor( 1 );
  _settextcolor ( BrightWhite );
  _clearscreen( _GCLEARSCREEN );

  OutlineScreen();

  _settextposition( 5, 10 );
  _outtext("Enter number of calibration points --> ");
  scanf("%d", &NumberOfCalPoints);
  _clearscreen( _GCLEARSCREEN );
```

```
  OutlineScreen();
  _settextposition( 4, 15 );
  _outtext("Connect C1 to 100 psi to set Calibration Mode ... ");
  _settextposition( 7, 25 );
  sprintf(buffer,"Calibration Point:      of %d", NumberOfCalPoints);
  _outtext( buffer );

  for( i = 0; i < NumberOfCalPoints; i++)
     {
     _settextposition( 7, 46 );
     sprintf(buffer,"%2d", i+1);
     _outtext( buffer );

     _settextposition( 10 + i, 10 );
     _outtext("Enter Pressure reading from BaroCell --> ");
     scanf("%f", &input);
     Pressure[i] = input;

     for ( ESP_PortNumber = 1; ESP_PortNumber <= NumberOfPorts;
                                                  ESP_PortNumber++ )
        Voltage[i][ESP_PortNumber-1] = Address_and_Sample();

     }

  Compute_Coefficients();

}

void Compute_Coefficients( void )
{
  int i, j, pv, row, col, Port;
  double a[3][3], r[3], r0, r1, r2;
  double sum_x, sum_x_squared, sum_x_cubed, sum_x_fourth;
  double sum_y, sum_xy, sum_yxx, d, pivot, factor;

  for ( Port = 0; Port < NumberOfPorts; Port++)
     {
     for ( i = 0; i < 3; i++)
        {
        for( j = 0; j < 3; j++ )
           a[i][j] = 0.0000000;

        r[i] = 0.0000000;
        }

     sum_x         = 0.00000;
     sum_x_squared = 0.00000;
     sum_x_cubed   = 0.00000;
     sum_x_fourth  = 0.00000;
     sum_y         = 0.00000;
     sum_xy        = 0.00000;
     sum_yxx       = 0.00000;
```

```
     for ( i = 0; i < NumberOfCalPoints; i++ )
        {
        sum_x         += Voltage[i][Port];
        sum_x_squared += Voltage[i][Port] * Voltage[i][Port];
        sum_x_cubed   += Voltage[i][Port] * Voltage[i][Port] *
Voltage[i][Port];
        sum_x_fourth  += Voltage[i][Port] * Voltage[i][Port]
                                          * Voltage[i][Port] *
Voltage[i][Port];

        sum_y         += Pressure[i];
        sum_xy        += Pressure[i] * Voltage[i][Port];
        sum_yxx       += Pressure[i] * Voltage[i][Port] * Voltage[i][Port];
        }

     a[0][0] = (double)(NumberOfCalPoints);
     a[0][1] = sum_x;
     a[0][2] = sum_x_squared;
     a[1][0] = sum_x;
     a[1][1] = sum_x_squared;
     a[1][2] = sum_x_cubed;
     a[2][0] = sum_x_squared;
     a[2][1] = sum_x_cubed;
     a[2][2] = sum_x_fourth;

     r[0]    = sum_y;
     r[1]    = sum_xy;
     r[2]    = sum_yxx;

//
// Solve the system of equations using Gauss-Jordan Method. Based on code
// from "FORTRAN 77 and Numerical Methods for Engineers" by G.J. Borse on
// Pgs. 416-417. Destroys both the a matrix and the r matrix.
//

     d = 1.0;

     for ( pv = 0; pv < 3; pv++)
        {
        if( fabs( a[pv][pv] ) < 1.0e-06 )
           {
           _clearscreen( _GCLEARSCREEN );
           printf("\n\n\n\t\tError! Zero Pivot detected during Matrix
Solve.\n");
           printf("\t\tESP probably not in Calibration Mode.\n");
           printf("\t\tProgram Terminating.\n\n");
           exit(0);
           }

        pivot = a[pv][pv];
        d     = d * pivot;
```

```
        for ( j = 0; j < 3; j++)
           a[pv][j] = a[pv][j] / pivot;

        r[pv]    = r[pv] / pivot;

        for( row = 0; row < 3; row++ )
          {
          if (row != pv )
             {
             factor = a[row][pv] / a[pv][pv];
             for( col = 0 ; col < 3; col++ )
                a[row][col] = a[row][col] - factor * a[pv][col];

             r[row] = r[row] - factor * r[pv];
             }
          }
        }

// Matrix Solution Complete.

     Calibration_Coeffs[0][Port] = r[0];
     Calibration_Coeffs[1][Port] = r[1];
     Calibration_Coeffs[2][Port] = r[2];
     }
}

void Freestream( void )
{
   int ESC_Flag, i;
   float Voltage, Pressure, Velocity;
   char ch, buffer[40];

   _setbkcolor( 1 );
   _settextcolor ( BrightWhite );

   _clearscreen( _GCLEARSCREEN );
   OutlineScreen();

   _settextcolor ( Green );
   _settextposition( 20, 21 );
   _outtext("Press ESC to begin taking data ...");
   _settextcolor ( BrightWhite );

   ESC_Flag = 0;

   while( !ESC_Flag )
      {
      ESP_PortNumber = 16;  // Assuming the Total pressure is on Port 16
      Voltage  = 0.000;
      for( i = 0; i < 15; i++ )
         Voltage  = Voltage + Address_and_Sample();

      Voltage  =   Voltage / 15.0;
```

```
      Pressure =   Calibration_Coeffs[0][15]
                 + Calibration_Coeffs[1][15] * Voltage
                 + Calibration_Coeffs[2][15] * Voltage * Voltage;
      Velocity = sqrt( 2.0 * Pressure * 133.35 / 1.1344 );

      _settextposition( 10, 26 );
      sprintf( buffer, "Dynamic Pressure:  %8.4f torr", Pressure );
      _outtext( buffer );
      _settextposition( 11, 30 );
      sprintf( buffer, "Tunnel Speed: %9.2f m/s", Velocity );
      _outtext( buffer );

      if( kbhit() )
         {
         ch = getch();
         if ( ch == 27 ) ESC_Flag = 1;
         if ( ch == 0 )  ch = getch();
         }
      }

   _clearscreen( _GCLEARSCREEN );
}



void Delay ( long Milliseconds )
{
// Attempts to create a Turbo Pascal-like delay for x milliseconds routine
//
  clock_t time_1, time_2;

  time_1 = clock();   // Returns the amount of time in milliseconds
  time_2 = clock();   // since the program has begun.

  while ( (time_2 - time_1) < Milliseconds )
    {
    time_2 = clock();
    }
}

int Round( float RealValue )
{
   int RoundedValue;

   RoundedValue = (int)( RealValue);
   if( (RealValue - (float)(RoundedValue)) > 0.50000 )
           RoundedValue++;

   return( RoundedValue );
}

void SetUpCounter( unsigned int Count )
```

```
{
   word Base;
   byte hi, lo;

   Base   = 816;

   outp( Base + 2, 0 ); // Set gate to 0, Close the gate
   Delay(1);

   hi = HI(Count);
   lo = LOW(Count);

   outp( Base + 7, 48 );
   Delay(1);
   outp( Base + 4, lo );
   Delay(1);
   outp( Base + 4, hi );
   Delay(1);
   outp( Base + 2, 16 ); // Open the gate
}

void BuildScreen( void )
{
   int i;
   char buffer[90];

   _setbkcolor( 1 );
   _settextcolor ( 15 );
   _clearscreen( _GCLEARSCREEN );

   OutlineScreen();

   _settextposition( 5, 0);
   _outtext( LeftSideDivider );

   for ( i = 1; i < 79; i++)
      _outtext( Horizontal );

   _outtext( RightSideDivider );

   _settextposition( 2, 29);
   _outtext("T h e      D y P P i R");
   _settextposition( 3, 17);
   _outtext("D a t a      A c q u i t i o n     S y s t e m");

   _settextposition( 7, 31);
   GetDate( date );
   _outtext( date );

   _settextposition( 9, 27);
   sprintf(buffer, "Data Filename:");
   _outtext( buffer );
   _settextposition( 9, 42);
```

```
   sprintf( buffer, "%s", filename );
   _outtext( buffer );
   _settextposition( 11, 27);
   sprintf( buffer, "Clock Delay: %u", NumberOfClockPulses );
   _outtext( buffer );


   _settextposition( 14, 31 );
   sprintf(buffer,"Ensemble:      of %d", NumberOfEnsembles);
   _outtext( buffer );

   _settextposition( 16, 26);
   _outtext("System Status:");

}

void OutlineScreen( void )
{
   int i;

   for ( i = 1; i < 80; i++)
      {
      _settextposition( 1, i );
      _outtext( Horizontal );
      _settextposition( 24, i );
      _outtext( Horizontal );
      }

   for ( i = 2; i < 25; i++ )
      {
      _settextposition( i, 0 );
      _outtext( Vertical );
      _settextposition( i, 80 );
      _outtext( Vertical );
      }

   _settextposition(24, 0);
   _outtext( LowerLeftCorner );

   _settextposition( 24, 80 );
   _outtext( LowerRightCorner );

   _settextposition(1, 0);
   _outtext( TopLeftCorner );

   _settextposition( 1, 80 );
   _outtext( TopRightCorner );

   _settextposition( 24, 24);
   _outtext(" ESM Fluid Mechanics Laboratory ");
}

void GetDate( char date_string[40]  )
```

```
{
   struct dosdate_t date;
   char day_year[10];

   _dos_getdate( &date );

   switch ( date.month )
      {
      case 1:
        strcpy( date_string, "January" );
        break;
      case 2:
        strcpy( date_string, "Feburary" );
        break;
      case 3:
        strcpy( date_string, "March" );
        break;
      case 4:
        strcpy( date_string, "April" );
        break;
      case 5:
        strcpy( date_string, "May" );
        break;
      case 6:
        strcpy( date_string, "June" );
        break;
      case 7:
        strcpy( date_string, "July" );
        break;
      case 8:
        strcpy( date_string, "August" );
        break;
      case 9:
        strcpy( date_string, "September" );
        break;
      case 10:
        strcpy( date_string, "October" );
        break;
      case 11:
        strcpy( date_string, "November" );
        break;
      case 12:
        strcpy( date_string, "December" );
        break;
   }

   strcat( date_string, " ");
   sprintf( day_year,"%u, %4d", date.day, date.year);
   strcat( date_string, day_year);

}

unsigned char HI( unsigned int input)
```

```
{
   unsigned char hi_byte;

   hi_byte = (unsigned char)( input >> 8 );
   return( hi_byte );
}

unsigned char LOW( unsigned int input)
{
   unsigned char lo_byte;

   lo_byte = (unsigned char)( (input << 8 ) >> 8 );
   return( lo_byte );
}
```

APPENDIX E: *"PRESSRED"* CODE LISTING

This appendix contains the code listing for the "Pressred" program, which was the primary data reduction program used in both the ESM Wind Tunnel and in the Virginia Tech Stability Wind Tunnel. Pressred reduced both the ESP data and the RC Electronics data, but very rarely at the same time. Memory restrictions for programs running under MS-DOS, prevented there being enough memory available to reduce both at the same time. Therefore, this version of the code has the section for the reduction of the RC Electronics data commented out, it is ready to reduce ESP data. The code could output the ESP data in a Stanford Graphics matrix file format or in an Axum column file format for easy importation into these programs. The code applies Chauvenet's criterion to the data as part of the data reduction.

```
//      PRESSRED.C   The DyPPiR Unsteady Pressure Reduction Code!
//         (c) 1994-7 Norman W. Schaeffler
//

// C includes
#include <graph.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <math.h>
#include <string.h>

// Project Includes

#define LowerLeftCorner "╚"
#define TopLeftCorner   "╔"

#define LowerRightCorner "╝"
#define TopRightCorner   "╗"

#define BottomOfDivider "╧"
```

```
#define TopOfDivider    "╤"

#define LeftSideDivider "╟"
#define RightSideDivider "╢"

#define Horizontal "─"
#define Vertical   "│"

#define CenterCross "┼"

#define Green 10
#define Red 12
#define BrightWhite 15
#define Blinking 16

#define Internal 0
#define External 1


// Defines
#define NoCursor  0x2000
#define Underline 0x0707
#define CursorHome   _settextposition( 23, 2 )
#define Status _settextposition( 22, 14 )
#define MaxEnsembles 20
#define Version   "11/30/97  - Sept. 96 Entry -- No RC Reduction"

// Function Protoypes
void  BuildScreen( void );
void  OutlineScreen( void );
void  DidYouKnow( void );

// Structures
struct
   {
    char date[40];
    int  NumberOfEnsembles;
    int  NumberOfChannels;
    int  NumberOfPorts;
    int  SamplingRate;
    int  Number_Of_Samples;
    int  ESP_SamplingRate;
    int  Number_Of_ESP_Samples;
   } info;

// Global Variables

         char filename[13], filename2[13];
         FILE *DataFile, *OutputRC, *OutputESP;
       double Calibration_Coeffs[3][32];
          int i, j, k, SGX_Flag, motion, First;
//
// The RC_Data, hi and lo arrays must be identical here and in Pressman!
```

```
//
          int RC_Data[1500];     // Must Match Number_Of_Samples
          int hi[310][32],lo[310][32];  // = ESP_SamplingRate * ESP_Duration
//        float RC_Voltage[1000][MaxEnsembles];
        float ESP_V[300][16][MaxEnsembles],Pressure[300][16];

void main( int argc, char *argv[] )
{
   int Port, m, Accept, GoodEnsembles, AnyNew, NumChar;
   float temp, Mean, StdDev, low, high, AoA;
   char GoodData[MaxEnsembles], EnsembleNumber[5], buffer[40];
   char *period_ptr;

   _settextcolor(15);
   _setbkcolor(0);
   _clearscreen( _GCLEARSCREEN );

   if ( argc == 4 )
      {
      strcpy( filename, argv[1] );
      SGX_Flag = atoi(argv[2]);
      motion   = atoi(argv[3]);
      }
   else
      {
      if ( argc == 3 )
         {
         strcpy( filename, argv[1] );
         SGX_Flag = 1;
         motion   = 0;
         }
      else
         {
         DidYouKnow();
         printf("   THE PROMPTS:\n");
         printf("   Enter the filename for the data  -->         ");
         scanf("%s", &filename);
         printf("   Enter the value for the SG flag  -->         ");
         scanf("%d", &SGX_Flag);
         printf("   Enter the motion number          -->         ");
         scanf("%d", &motion);
         }
      }

   if( SGX_Flag > 1 ) SGX_Flag = 1;

   if ( (DataFile = fopen(filename,"rb")) == NULL )
     {
     _clearscreen( _GCLEARSCREEN );
     printf("Error opening datafile %s\nProgram Aborted\n", filename);
     exit(0);
     }
```

```
// Try to create the base for the filename

   strcpy( filename2, "");
   period_ptr = strchr(filename,'.');
   NumChar = period_ptr - &filename[1] + 1;
   strncat(filename2, filename, NumChar);
   if ( SGX_Flag )
      strcat( filename2, ".dat" );
   else
      strcat( filename2, ".esp" );

   printf("\n\n\n     The output file name is: %s\n", filename2);

   if ( (OutputESP = fopen(filename2,"w")) == NULL )
     {
     _clearscreen( _GCLEARSCREEN );
     printf("Error opening ESP output file %s\nProgram Aborted\n",
filename2);
     exit(0);
     }

   fread( &info, sizeof( info ), 1, DataFile );
   fread( Calibration_Coeffs, sizeof( Calibration_Coeffs ), 1, DataFile);

   _settextcursor( NoCursor );
   BuildScreen();

   _settextposition( 5, 5);
   sprintf(buffer, "Data contained in %s was acquired on %s.",
                    filename, info.date);
   _outtext(buffer);

   _settextposition( 8, 5);
   sprintf(buffer, "There were %3d ensembles of data taken.",
                    info.NumberOfEnsembles);
   _outtext(buffer);

   _settextposition( 10, 5);
   sprintf(buffer, "The RC Data consists of %2d channels sampled at %4d Hz",
                     info.NumberOfChannels, info.SamplingRate );
   _outtext(buffer);
   _settextposition( 11, 5);
   sprintf(buffer, "With a total of %5d samples taken.",
                     info.Number_Of_Samples );
   _outtext(buffer);

   _settextposition( 13, 5);
   sprintf(buffer, "The ESP Data consists of %2d ports sampled at %4d Hz",
                     info.NumberOfPorts, info.ESP_SamplingRate );
   _outtext(buffer);
   _settextposition( 14, 5);
   sprintf(buffer, "With a total of %5d samples taken per port.",
                     info.Number_Of_ESP_Samples );
```

```
   _outtext(buffer);

   _settextposition( 16, 5);
   if ( SGX_Flag )
      sprintf(buffer, "Output File Format :> Stanford Graphics % d", motion);
   else
      sprintf(buffer, "Output File Format :> Axum.");

   _outtext(buffer);

   _settextposition( 18, 5);
   _outtext("PRESSRED Version:");
   _settextposition( 18, 23);
   _outtext( Version );

   _settextposition( 22, 5 );
   _outtext("Status: ");

//   while ( !kbhit() )   {}

   for( i = 0; i < info.NumberOfEnsembles; i++)
      {
      Status;
      sprintf(buffer,"Reading dataset %3d of %3d", i+1,
                                                   info.NumberOfEnsembles);
      _outtext( buffer );


      fread( hi, sizeof( hi ), 1, DataFile );
      fread( lo, sizeof( lo ), 1, DataFile );
      fread( RC_Data, sizeof( RC_Data ), 1, DataFile );

//      for ( j = 0; j < info.Number_Of_Samples; j++)
//         RC_Voltage[j][i] = (RC_Data[j] - 2048) * 0.004883;

      for ( j = 0; j < info.Number_Of_ESP_Samples; j++)
         {
         for ( k = 0; k < info.NumberOfPorts; k++)
            ESP_V[j][k][i] = ( 16 * hi[j][k] + lo[j][k]/16 )
                                                   * 10.0 / 4096 - 5.0;

         }
      }

//   Status;
//   _outtext("Processing and Writing RC data ...");
//   for( i = 0; i < info.NumberOfEnsembles; i++)
//      {
//      strcpy( filename2, basename );
//      if( i < 9 )
//         sprintf(EnsembleNumber,".00%1d",i+1);
//      else
//         sprintf(EnsembleNumber,".0%2d",i+1);
```

```
//      strcat( filename2, EnsembleNumber );
//
//      if ( (OutputRC = fopen(filename2,"w")) == NULL )
//        {
//        _clearscreen( _GCLEARSCREEN );
//        printf("Error opening file %s\nProgram Aborted\n", filename2);
//        exit(0);
//        }
//
//     for ( j = 0; j < 16; j+4)
//        fprintf(OutputRC, " %3d %6.3f %6.3f %6.3f %6.3f\n",
//                j, RC_Voltage[j][i], RC_Voltage[j+1][i],
//                   RC_Voltage[j+2][i], RC_Voltage[j+3][i]);
//
//      fclose( OutputRC );
//      }

   Status;
   _outtext("Processing ESP data ...              ");
   for ( j = 0; j < info.Number_Of_ESP_Samples; j++)
      {
      for( k = 0; k < info.NumberOfPorts; k++ )
        {
        Accept = 0;
        GoodEnsembles = info.NumberOfEnsembles;
        for( m = 0; m < MaxEnsembles; m++)
           GoodData[m] = 1;

        while ( Accept == 0 )
           {
           Mean   = 0.0000;
           StdDev = 0.0000;

           for( m = 0; m < info.NumberOfEnsembles; m++)
              {
              if ( GoodData[m] )
                {
                Mean   += ESP_V[j][k][m];
                StdDev += ESP_V[j][k][m] * ESP_V[j][k][m];
                }
              }

           if( GoodEnsembles == 0 )
              {
              Mean = 0.000;
              break;
              }

           Mean = Mean / GoodEnsembles;
           StdDev = sqrt( StdDev / GoodEnsembles - Mean*Mean );

// The constant multiplying the StdDev is based on 20 ensembles!!!!!
```

```
            high   = Mean + 2.23 * StdDev;
            low    = Mean - 2.23 * StdDev;
            AnyNew = 0;

            for( m = 0; m < info.NumberOfEnsembles; m++)
               {
               if ( GoodData[m] )
                  {
                  if( ( ESP_V[j][k][m] > high ) || ( ESP_V[j][k][m] < low ))
                     {
                     GoodData[m] = 0;
                     GoodEnsembles = GoodEnsembles - 1;
                     AnyNew = 1;
                     }
                  }
               }

            if ( AnyNew == 0 )
               Accept = 1;

            }

         Mean = Mean + 0.012 + 0.00346478 * ( 4.064 - Mean );
         Pressure[j][k] =    Calibration_Coeffs[0][k]
                           + Calibration_Coeffs[1][k] * Mean
                           + Calibration_Coeffs[2][k] * Mean * Mean;
         }
      }

   Status;
   _outtext("Writing ESP data to file ...      ");

   if ( motion == 1 ) First = 19;
   else if ( motion == 2 ) First = 39;
   else if ( motion == 3 ) First = 39;
   else First = 0;

   if( SGX_Flag )
      {
// Old News: Left for posterity
// This 200 is a hard code for this particular set of runs. 200 samples/sec
// The follwing line is the matrix dimensions, also hard coded.
// Also the delta-alpha per time step is motion specific.

      if ( motion == 1 ) fprintf(OutputESP, "280 15 \n");
      else if ( motion == 2 ) fprintf(OutputESP, "260 15 \n");
      else if ( motion == 3 ) fprintf(OutputESP, "260 15 \n");
      else fprintf(OutputESP, "%d 15 \n", info.Number_Of_ESP_Samples);

      fprintf(OutputESP, "NaN");
      for ( j = 0; j < info.Number_Of_ESP_Samples; j++)   {
         AoA = (float)j / (float)info.ESP_SamplingRate;  // This is really
time now
```

```
         if ( j > First ) fprintf(OutputESP, " %9.5f", AoA);
         j++;
         }

      fprintf(OutputESP, "\n");

      for( i = 0; i < info.NumberOfPorts-1; i++)  {
         fprintf(OutputESP, " %2d", i-7);
         for ( j = 0; j < info.Number_Of_ESP_Samples; j++)   {
            if ( j > First )
               fprintf (OutputESP, " %9.5f", Pressure[j][i]/Pressure[j][15]);
            j++;
            }

         fprintf(OutputESP, "\n");
         }
      }
   else  {    // This section is not "fixed" for motions
      for( i = 0; i < info.NumberOfPorts-1; i++)   {
         fprintf(OutputESP, " %2d", i+1);
         for ( j = 0; j < info.Number_Of_ESP_Samples; j++)
           fprintf(OutputESP, " %9.5f", Pressure[j][i]/Pressure[j][15]);

         fprintf(OutputESP, "\n");
         }
      }


   fclose( DataFile );
   fclose( OutputESP );

   Status;
   _outtext("Complete.                          ");


   _settextcursor( Underline );
   _setbkcolor( 0 );
   _settextcolor ( 15 );
   _clearscreen( _GCLEARSCREEN );
   _clearscreen( _GCLEARSCREEN );

}

void BuildScreen( void )
{
   int i;
   char buffer[90];

   _setbkcolor( 1 );
   _settextcolor ( 15 );
   _clearscreen( _GCLEARSCREEN );

   OutlineScreen();
```

```
   _settextposition( 3, 0);
   _outtext( LeftSideDivider );

   for ( i = 1; i < 79; i++)
      _outtext( Horizontal );

   _outtext( RightSideDivider );

   _settextposition( 2, 25);
   _outtext("Dynamic Pressure Data Reduction");

}

void OutlineScreen( void )
{
   int i;

   for ( i = 1; i < 80; i++)
      {
      _settextposition( 1, i );
      _outtext( Horizontal );
      _settextposition( 24, i );
      _outtext( Horizontal );
      }

   for ( i = 2; i < 25; i++ )
      {
      _settextposition( i, 0 );
      _outtext( Vertical );
      _settextposition( i, 80 );
      _outtext( Vertical );
      }

   _settextposition(24, 0);
   _outtext( LowerLeftCorner );

   _settextposition( 24, 80 );
   _outtext( LowerRightCorner );

   _settextposition(1, 0);
   _outtext( TopLeftCorner );

   _settextposition( 1, 80 );
   _outtext( TopRightCorner );

   _settextposition( 24, 24);
   _outtext(" ESM Fluid Mechanics Laboratory ");

}

void DidYouKnow( void )
{
```

```
  printf("                  M A N U A L    E N T R Y    M O D E    \n\n\n");
  printf("        Did you know that you could have entered the input file
name\n");
  printf("        right on the command line?\n\n");
  printf("                                AVOID THE PROMPTS!\n\n");
  printf("        Usage: C:> pressred inputfile SGX_Flag\n\n");
  printf("           Just another wacky time saving convenience from
Shi*Ware!\n\n\n");
}
```

VITA

Norman Walter Schaeffler was born the first son and eldest child of Judith Anita Saul and Norman Charles Schaeffler on December 10, 1965 in Smithtown, New York. He is the third in a line of Norman Schaefflers, the first being his late grandfather, Norman Wilfred Schaeffler. He attended Floyd E. Kellam High School in Virginia Beach, Virginia, where he graduated 10th, and in the top 3%, of his class. He graduated with a Bachelor of Science in Engineering Science and Mechanics from Virginia Polytechnic Institute and State University in 1987, where he received the T. Marshall Hann Scholarship. He received his Master of Science in Engineering Mechanics from VPI&SU in 1988. He subsequently spent three years in the employment of Newport News Shipbuilding and Dry Dock Company as an engineer in the Seawolf Machinery group (E14), designing components for the SEAWOLF class U.S. Navy nuclear attack submarine. In 1992, he matriculated to VPI&SU to persue a Ph.D. in Engineering Mechanics.

On a beautiful fall day in September of 1991, Norman Walter Schaeffler married Louise Carlyn Miller in Woodstock, Virginia. On another beautiful fall day in October of 1996, Louise Carlyn Schaeffler gave birth to Carlyn Anita Schaeffler, the apple of her father's eye.

Norman W. Schaeffler's other interests include golf, American history and pre-history and archeology. He is a member of the Archeology Society of Virginia.

Norman W. Schaeffler

April, 1998
Blacksburg, Virginia

"All honor to Jefferson, to the man who, in the concrete pressure of a struggle for national independence by a single people, had the coolness, forecast, and capacity to introduce into a merely revolutionary document, an abstract truth, applicable to all men and all times, and so to embalm it there, that to-day, and in all coming days, it shall be a rebuke and a stumbling block to the very harbingers of re-appearing tyranny and oppression."

-Abraham Lincoln

"I can never stand still. I must explore and experiment. I am never satisfied with my work. I resent the limitations of my own imagination."

-Walter Elias Disney

A NOTE ON THE TYPE

This document uses a modern adaptation of the Caslon font. The typeface is named for its originator, William Caslon (1692-1766). The typeface was very popular in the eighteenth century. Examples of where this typeface was used include the first printed currency of the United States and the first printed copies of the Declaration of Independence and the U. S. Constitution. The typeface remains popular today. This concludes the history lesson for today.